Multilevel methods in optimization with partial differential equations

Alfio Borzì

Institut für Mathematik und Wissenschaftliches Rechnen
Karl–Franzens–Universität Graz
Heinrichstraße 36, 8010 Graz, Austria

Phone:+43 316 380 5166
Fax: +43 316 380 9815
www: http://www.uni-graz.at/imawww/borzi/index.html
email: alfio.borzi@uni-graz.at

Contents

1 Introduction

In these Lecture Notes we give an introduction to advanced multilevel strategies for solving unconstrained and constrained optimization problems governed by partial differential equations (PDE). On the one hand, practical aspects for the development and validation of multilevel algorithms for PDE-model-based optimization are discussed. On the other hand, recent convergence analysis results for some representative multilevel optimization schemes are presented.

These lecture notes start with an introduction of classical multilevel (multigrid) schemes for linear and nonlinear scalar elliptic problems that also correspond to unconstrained optimization problems as minimization of appropriate energy functionals. This fact suggests that we may interpret well-known multilevel schemes as optimization methods per se. Along this line, we discuss the MG/OPT approach to unconstrained optimization problems and related convergence analysis results.

The multilevel strategy is also successful in solving optimization problems with PDE constraints as they appear, for example, in the formulation of optimal control problems. Within this framework, we focus on one-shot multilevel schemes. For these methods, convergence estimates are also discussed.

Indeed, classical multilevel solvers can be used in combination with gradient-type and Newton-like optimization methods. However, in these cases no special use of inherent properties of the multilevel strategy is made.

With multilevel strategy we mean a methodology of viewing a problem involving many different length-scales in such a way to treat each length-scale efficiently on an appropriately chosen space and to combine the results of this treatment to obtain a fast and accurate solution to the original problem.

For the purpose of intuition, we give now a few examples of 'multiscale phenomena' or 'multilevel phenomena'. Specifically, we would

like to show that there are systems having many degrees of freedom, and are defined on many different length-scales, that possess a special structure such that they are essentially invariant under length-scaling.

A first example is percolation (Orig. Latin: to filter through). Percolation deals with effects related to the varying of the richness of interconnections in a infinite network system. For example, consider the electric network of Figure 1. If the fraction of connecting links is higher than some transition value p^*, the net is conducting. Below this value, the net is insulating.

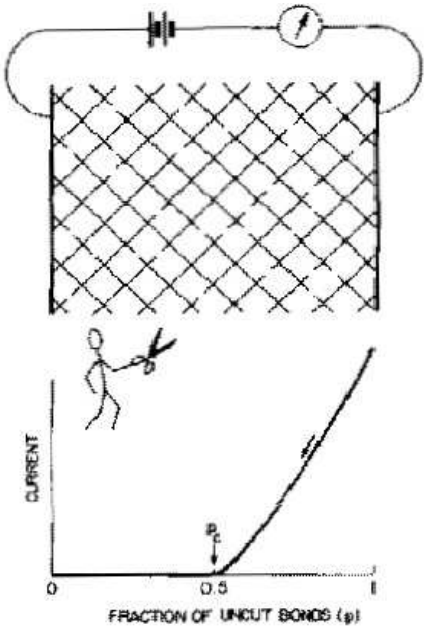

Figure 1: Electric network.

To have an insight on this transition phenomenon, consider the simple case of vertical percolation. In a 2×2 box there is vertical percolation in the cases depicted in Figure 2, which correspond to the occurrence of a continuous dark column. Assume that a single square in the box percolates (is dark) with probability p. Then the probability that the 2×2 cell percolates is

$$p' = R(p) = 2p^2(1-p^2) + 4p^3(1-p) + p^4$$

The mapping $p \rightarrow R(p)$ has a fixed point, $p^* = R(p^*)$, which is the

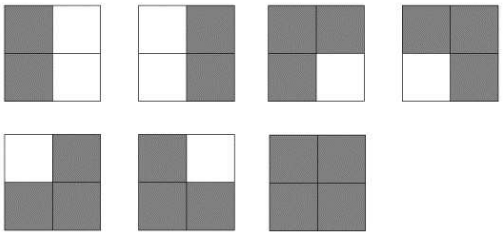

Figure 2: Vertical percolation in a 2×2 box.

critical value p^* to have percolation; in our case $p^* = 0.6180$. Now define a coarsening (scaling) procedure, applied to a $n \times n$ (ideally infinite) box, which consists in replacing a 2×2 percolating box with a percolating square, otherwise a non percolating square. If the original box is characterized by $p < p^*$, the repeated application of the coarsening procedure results in a 'visible' non percolating box; see Figure 3.

In the other case of $p > p^*$, a repeated application of the coarsening

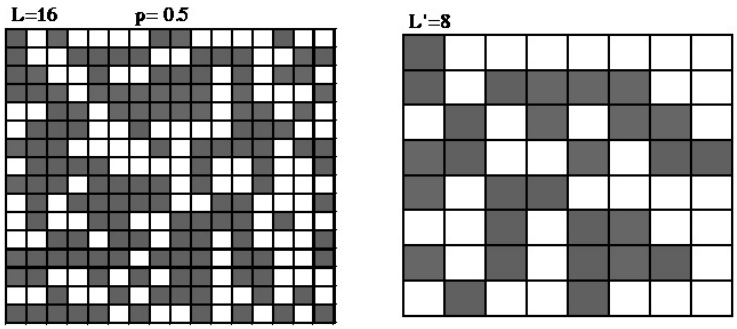

Figure 3: One coarsening step: No percolation ($p < p^*$).

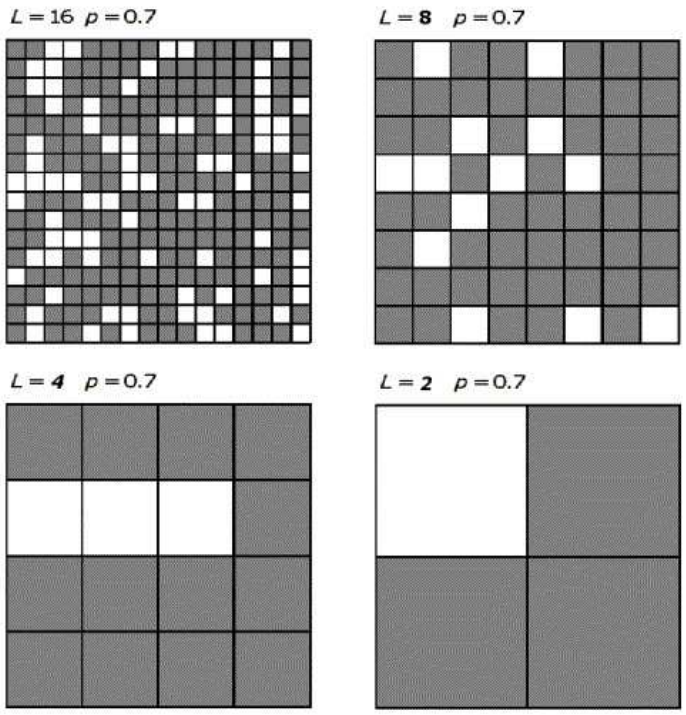

Figure 4: Four coarsening steps: Percolation ($p > p^*$).

process reveals percolation as illustrated in Figure 4. At $p = p^*$ the distribution of connections is such that we have 'coarsening or scaling invariance'.

A second example that provides algebraic insight in the multilevel approach is cyclic reduction. Consider the finite-difference discretization of $-u'' = f$ on some interval and with homogeneous Dirichlet boundary conditions. On a uniform grid with 7 interior points the corresponding algebraic problem is given by the following linear system

$$\begin{array}{c} E_1 \\ E_2 \\ E_3 \\ E_4 \\ E_5 \\ E_6 \\ E_7 \end{array} \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = h^2 \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}$$

Let us call the equation corresponding to the row i as E_i. We define the following coarsening steps

$$E_i' = R(E_i) = E_{i-1} + 2\,E_i + E_{i+1}, \quad i = 2, 4, 6.$$

The result of one coarsening is

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \cdot \begin{bmatrix} u_2 \\ u_4 \\ u_6 \end{bmatrix} = H^2 \begin{bmatrix} \tilde{f}_2 \\ \tilde{f}_4 \\ \tilde{f}_6 \end{bmatrix} \quad (H = 2h),$$

where $\tilde{f}_i = (f_{i-1} + 2f_i + f_{i+1})/4$. Thus we obtain a smaller algebraic system with the same structure as the original one. In this sense we have 'coarsening invariance'. We can start appreciating the advantage of a system possessing coarsening invariance: We can re-solve properties of the system considering its coarsened version. In the first example, we can state the percolation property of the network by examining a coarsened version of the network. In the second example, we can compute part of the entire solution solving a system of 3 unknowns instead of a system of 7 unknowns. And this process can be repeated recursively.

The second example allows also to make a direct link to unconstrained optimization as minimization of an appropriate functional. Notice that the matrix of coefficients of the 7 unknown linear system is symmetric and positive definite. Denote this system with $Au = f$. It

is well known that the solution $u = A^{-1}f$ is the only minimizer of the functional (T means transpose)

$$J(u) = \frac{1}{2}\, u^T A u - u^T f.$$

Similarly, if $\tilde{A}\tilde{u} = \tilde{f}$ denotes the 3 unknown system given above, we have that the minimum of

$$\tilde{J}(\tilde{u}) = \frac{1}{2}\, \tilde{u}^T \tilde{A} \tilde{u} - \tilde{u}^T \tilde{f}$$

corresponds to the minimum of J in the subspace of components with even index. Thus we have the possibility to approach the problem of minimizing the functional J defined on a given finite dimensional space by considering the minimization of another functional $\tilde{J}$ defined on a smaller space.

An essential concept that motivates the development of multilevel methods is the concept of computational complexity. Consider a system described by n unknowns defining the solution of an algebraic system or the minimum of a configuration functional. A best possible (optimal) solver is one that provides this solution with a number of operations which is linearly proportional to n. We shall demonstrate that in most cases algorithms based on the multilevel strategy result in optimal solvers.

We conclude this section with some historical remarks and references. Already in the sixties R.P. Fedorenko [21, 22] developed the first multilevel scheme for the solution of the Poisson equation in a unit square. Since then, other mathematicians extended Fedorenko's idea to general elliptic boundary value problems with variable coefficients; see, e.g., [1]. However, the full efficiency of the multilevel approach was realized after the works of A. Brandt [13, 14] and W. Hackbusch [24]. These Authors also introduced multilevel methods for nonlinear problems like the multilevel *full approximation storage* (FAS) scheme [14, 27]. Another achievement in the formulation of multilevel methods

was the *full multigrid* (FMG) scheme [14, 27], based on the combination of nested iteration techniques and multilevel methods. Multilevel algorithms are now applied to a wide range of problems, primarily to solve linear and nonlinear boundary value problems. Other examples of successful applications are eigenvalue problems [15, 25], bifurcation problems [37, 46], parabolic problems [18, 26, 50], hyperbolic problems [19, 38], and mixed elliptic/hyperbolic problems. Investigation of multilevel methods for solving PDE-based optimization problems is more recent, see, e.g., [47]. Also convergence analysis of multilevel methods has developed along with the multitude of applications.

An essential contribution to the development of the multilevel community is the MGNet of Craig C. Douglas. This is the communication platform on everything related to multilevel methods;
see http://www.mgnet.org.

Classical Readings

1. J.H. Bramble, *Multigrid Methods*, Pitman research notes in mathematical series, Longman Scientific & Technical, 1993.

2. A. Brandt, *Multi-grid techniques: 1984 guide with applications to fluid dynamics*, GMD-Studien. no 85, St. Augustin, Germany, 1984.

3. W.L. Briggs, *A Multigrid Tutorial*, SIAM, Philadelphia, 1987.

4. W. Hackbusch, *Multi-Grid Methods and Applications*, Springer-Verlag, Heidelberg, 1985.

5. S.F. McCormick, *Multigrid Methods*, Frontiers in Applied Mathematics, vol. 3, SIAM Books, Philadelphia, 1987.

6. U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.

7. P. Wesseling, *An introduction to multigrid methods*, John Wiley, Chichester, 1992.

2 Multilevel methods for linear problems

The basic components of a multilevel algorithm are presented in this section. We start with the analysis of two standard iterative techniques: the Jacobi and Gauss-Seidel schemes. These two methods are characterized by global poor convergence rates, however, for errors whose length scales are comparable to the mesh size, they provide rapid damping, leaving behind smooth, longer wave-length errors. These smooth components are responsible for the slow global convergence. A multilevel algorithm, employing grids of different mesh sizes, allows to solve all wave-length components and provides rapid convergence rates. The multilevel strategy combines two complementary schemes. The high-frequency components of the error are reduced applying iterative methods like Jacobi or Gauss-Seidel schemes. For this reason these methods are called smoothers. On the other hand, low-frequency error components are effectively reduced by a coarse-grid correction procedure. Because the action of a smoothing iteration leaves only smooth error components, it is possible to represent these components as the solution of an appropriate coarser system. Once this coarser problem is solved, its solution is interpolated back to the fine grid to correct the fine grid approximation for the low-frequency errors.

2.1 Iterative methods and the smoothing property

Consider a large sparse linear system of equations $Au = f$, where A is a symmetric positive $n \times n$ matrix. Iterative methods for solving this problem are formulated as follows

$$u^{(\nu+1)} = Mu^{(\nu)} + Nf \ , \tag{1}$$

where M and N have to be constructed in such a way that given an arbitrary initial vector $u^{(0)}$, the sequence $u^{(\nu)}$, $\nu = 0, 1, \ldots$, converges to the solution $u = A^{-1}f$. Define the solution error at the sweep ν as

$e^{(\nu)} = u - u^{(\nu)}$, then the iteration (1) is equivalent to $e^{(\nu+1)} = Me^{(\nu)}$. M is called the *iteration matrix*. We have the following convergence criterion based on the *spectral radius* $\rho(M)$ of the matrix M.

Theorem 1 *The method (1) converges if and only if* $\rho(M) < 1$.

A general framework to define iterative schemes of the type (1) is based on the concept of *splitting*. Assume the splitting $A = B - C$ where B is non singular. By setting $Bu^{(\nu+1)} - Cu^{(\nu)} = f$ and solving with respect to $u^{(\nu+1)}$ one obtains

$$u^{(\nu+1)} = B^{-1}Cu^{(\nu)} + B^{-1}f \ .$$

Thus $M = B^{-1}C$ and $N = B^{-1}$. Typically, one considers the regular splitting $A = D - L - U$ where $D = \mathrm{diag}(a_{11}, a_{22}, \ldots, a_{nn})$ denotes the diagonal part of the matrix A, and $-L$ and $-U$ are the strictly lower and upper parts of A, respectively. Based on this splitting many choices for B and C are possible leading to different iterative schemes. For example, the choice $B = \frac{1}{\omega}D$ and $C = \frac{1}{\omega}[(1-\omega)D + \omega(L+U)]$, $0 < \omega \leq 1$, leads to the *damped* Jacobi iteration

$$u^{(\nu+1)} = (I - \omega D^{-1}A)u^{(\nu)} + \omega D^{-1}f \ . \tag{2}$$

Choosing $B = D - L$ and $C = U$, one obtains the Gauss-Seidel iteration

$$u^{(\nu+1)} = (D-L)^{-1}Uu^{(\nu)} + (D-L)^{-1}f \ . \tag{3}$$

Later on we denote the iteration matrices corresponding to (2) and (3) with $M_J(\omega)$ and M_{GS}, respectively.

To define and analyze the smoothing property of these iterations we introduce a simple model problem. Consider the finite-difference approximation of a one-dimensional Dirichlet boundary value problem:

$$\begin{cases} -\frac{d^2u}{dx^2} = f(x), & \text{in} \quad \Omega = (0,1) \\ u(x) = g(x), & \text{on} \quad x \in \{0,1\} \ . \end{cases} \tag{4}$$

Let Ω be represented by a grid Ω_h with grid size $h = \frac{1}{n+1}$ and grid points $x_j = jh$, $j = 0, 1, \ldots, n+1$. A discretization scheme for the second derivative at x_j is $h^{-2}[u(x_{j-1}) - 2u(x_j) + u(x_{j+1})] = u''(x_j) + O(h^2)$. Set $f_j^h = f(jh)$, and $u_j^h = u(jh)$. We obtain the following tridiagonal system of n equations

$$\begin{aligned} 2\,u_1^h - u_2^h &= h^2\,f_1^h + g(0) \\ -u_{j-1}^h + 2\,u_j^h - u_{j+1}^h &= h^2\,f_j^h, \qquad j = 2, \ldots, n-1 \\ -u_{n-1}^h + 2\,u_n^h &= h^2\,f_n^h + g(1) \end{aligned} \tag{5}$$

Let us denote (5) (with all terms divided by h^2) by $A_h\,u_h = f_h$.

We discuss the solution to this problem by means of the damped Jacobi iteration with iteration matrix $M_J(\omega) = I - \omega D_h^{-1} A_h$. Consider the eigenvalue problem $M_J(\omega)v^k = \mu_k v^k$. The eigenvectors of $M_J(\omega)$ (and equivalently of A_h) are given by

$$v^k = \sqrt{2h}\,(\sin(k\pi h j))_{j=1,n} \;, \quad k = 1, \ldots, n \;, \tag{6}$$

The eigenvalues of A_h are $\lambda_k = 4\,\sin^2(k\pi h/2)/h^2$ and the corresponding eigenvalues of $M_J(\omega)$ are

$$\mu_k(\omega) = 1 - \omega(1 - \cos(k\pi h)) \;, \quad k = 1, \ldots, n \;. \tag{7}$$

We have that $\rho(M_J(\omega)) < 1$ for $0 < \omega \leq 1$, guaranteeing convergence. In particular, for the Jacobi iteration with $\omega = 1$ we have $\rho(M_J(1)) = 1 - \frac{1}{2}\pi h^2 + O(h^4)$, showing how the convergence of the Jacobi iteration deteriorates (i.e. ρ tends to 1) as $h \to 0$.

The purpose of an iteration in a MG algorithm is primarily to be a smoothing operator. In order to define this property, we need to distinguish between low- and high-frequency eigenvectors. We define

- *low frequency* (LF) components: v^k with $1 \leq k < \frac{n}{2}$;
- *high frequency* (HF) components: v^k with $\frac{n}{2} \leq k \leq n$;

We now define the *smoothing factor* μ as the worst factor by which the amplitudes of HF components are damped per iteration. In the case of the Jacobi iteration we have

$$\begin{aligned}\mu &= \max\{|\mu_k|, \frac{n}{2} \le k \le n\} = \max\{1-\omega,\ |1-\omega(1-\cos(\pi)|\} \\ &\le \max\{1-\omega, |1-2\omega|\}.\end{aligned}$$

Using this result we find that the optimal (smallest) smoothing factor $\mu = 1/3$ is obtained choosing $\omega^* = 2/3$. This means that using $M_J(\omega^*)$ the HF error components are reduced at least of a factor three after any iteration sweep and this factor does not depend on the mesh size. Therefore if we use the expansion $e^{(\nu)} = \sum_k e_k^{(\nu)} v^k$, we have that a few sweeps of (2) give $|e_k^{(\nu)}| << |e_k^{(0)}|$ for HF components. For this reason, although the global error decreases slowly by iteration, it is smoothed very quickly.

Most often, instead of a Jacobi method other iterations are used that suppress the HF components of the error more efficiently. This is the case of the Gauss-Seidel iteration (3). The smoothing property of this scheme is conveniently analyzed by using *local mode analysis* (LMA) introduced by Brandt [14]. This is an effective tool for analyzing the MG process even though it is based on certain idealized assumptions and simplifications: Boundary conditions are neglected and the problem is considered on infinite grids $G^h = \{jh,\ j \in \mathbb{Z}\}$. Notice that on G^h, only the components $e^{i\theta x/h}$ with $\theta \in (-\pi, \pi]$ are visible, i.e., there is no other component with frequency $\theta_0 \in (-\pi, \pi]$ with $|\theta_0| < \theta$ such that $e^{i\theta_0 x/h} = e^{i\theta x/h}$, $x \in G^h$.

In local Mode (Fourier) analysis the notion of low- and high-frequency components on the grid G^h is related to a coarser grid denoted by G^H. In this way $e^{i\theta x/h}$ on G^h is said to be an high-frequency component, with respect to the coarse grid G^H, if its restriction (projection) to G^H is not visible there. If $H = 2h$ then the high frequencies are those with $\frac{\pi}{2} \le |\theta| \le \pi$. We have $e^{i\theta x/h} = e^{i(2\theta)x/H}$.

In this framework, in order to analyze a given iteration we represent solution errors in terms of their θ components $e^{(\nu)} = \sum_\theta \mathcal{E}_\theta^{(\nu)} e^{i\theta x/h}$ and $e^{(\nu+1)} = \sum_\theta \mathcal{E}_\theta^{(\nu+1)} e^{i\theta x/h}$ (with formal summation on θ). Where $\mathcal{E}_\theta^{(\nu)}$ and $\mathcal{E}_\theta^{(\nu+1)}$ denote the error amplitudes of the θ component, before and after smoothing, respectively. The action of the iteration matrix M is $e^{(\nu+1)} = M\, e^{(\nu)}$. In the Fourier space this action is represented by $\mathcal{E}_\theta^{(\nu+1)} = \hat{M}(\theta)\, \mathcal{E}_\theta^{(\nu)}$, and $\hat{M}(\theta)$ is the Fourier symbol of M.

In the LMA framework, the smoothing factor is then defined by

$$\mu = \max\{|\frac{\mathcal{E}_\theta^{(\nu+1)}}{\mathcal{E}_\theta^{(\nu)}}| \ , \ \frac{\pi}{2} \leq |\theta| \leq \pi\} = \max\{|\hat{M}(\theta)| \ , \ \frac{\pi}{2} \leq |\theta| \leq \pi\} \ . \tag{8}$$

Later, we consider the entire frequency domain spanned by the two sets of frequencies $\theta \in ([-\pi/2, \pi/2)$ and $\overline{\theta} = \theta - sign(\theta)\pi$. Here θ represents low frequencies components while $\overline{\theta}$ contains the high frequencies components. This choice results in the basis of the two harmonics $e^{i\theta x/h}$ and $e^{i\overline{\theta} x/h}$. In this framework, a way to characterize the smoothing property of the smoothing operator M is to consider the action of M on both sets of frequencies as follows

$$\widehat{M}(\theta) = \begin{bmatrix} \hat{M}(\theta) & 0 \\ 0 & \hat{M}(\overline{\theta}) \end{bmatrix}$$

and to assume an ideal coarse grid correction which annihilates the low frequency error components and leaves the high frequency error components unchanged. That is, one defines the projection operator $\widehat{Q}$ as follows

$$\widehat{Q}(\theta) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

In this framework the smoothing property of M is defined as follows

$$\mu = \max\{r(\widehat{Q}(\theta)\, \widehat{M}(\theta)) : \theta \in [-\pi/2, \pi/2)\}, \tag{9}$$

where r is the spectral radius.

For illustration consider the Gauss-Seidel scheme applied to our discretized model problem. A smoothing sweep starting with an initial approximation $u^{(\nu)}$ produces a new approximation $u^{(\nu+1)}$ such that the corresponding error satisfies

$$B_h e^{(\nu+1)}(x) - C_h e^{(\nu)}(x) = 0, \quad x \in G^h, \tag{10}$$

where $B_h = D_h - L_h$ and $C_h = U_h$. For a given θ, equation (10) at $x = jh$ becomes

$$\sum_{\theta} [(2 - e^{-i\theta})\mathcal{E}_\theta^{(\nu+1)} - e^{i\theta}\mathcal{E}_\theta^{(\nu)}]\, e^{i\theta j} = 0$$

which must hold for all j, therefore we obtain $\hat{M}(\theta) = e^{i\theta}/(2 - e^{-i\theta})$. We have

$$\mu = \max\{|\frac{\mathcal{E}_\theta^{(\nu+1)}}{\mathcal{E}_\theta^{(\nu)}}|,\ \frac{\pi}{2} \le |\theta| \le \pi\} = \max\{|\frac{e^{i\theta}}{2 - e^{-i\theta}}|,\ \frac{\pi}{2} \le |\theta| \le \pi\} = 0.45.$$

Similar values are obtained for the Gauss-Seidel iteration applied to the two- and three-dimensional version of our model problem. For the two dimensional case the effect of smoothing can be seen in Figure 5.

Another definition of smoothing property of an iterative scheme is due to Hackbusch [27]. Let M be the iteration matrix of a smoothing procedure and recall the relation $e^{(\nu)} = M^\nu e^{(0)}$. One can measure the smoothness of $e^{(\nu)}$ by a norm involving differences of the value of this error on different grid points. A natural choice is to take the second-order difference matrix A above (without multiplication by h^2). Then the following smoothing factor is defined

$$\mu(\nu) = \|AM^\nu\|/\|A\|.$$

The iteration defined by M is said to possess the smoothing property if there exists a function $\eta(\nu)$ such that, for sufficiently large ν, we have

$$\|AM^\nu\| \le \eta(\nu)\, h^{-\alpha}, \tag{11}$$

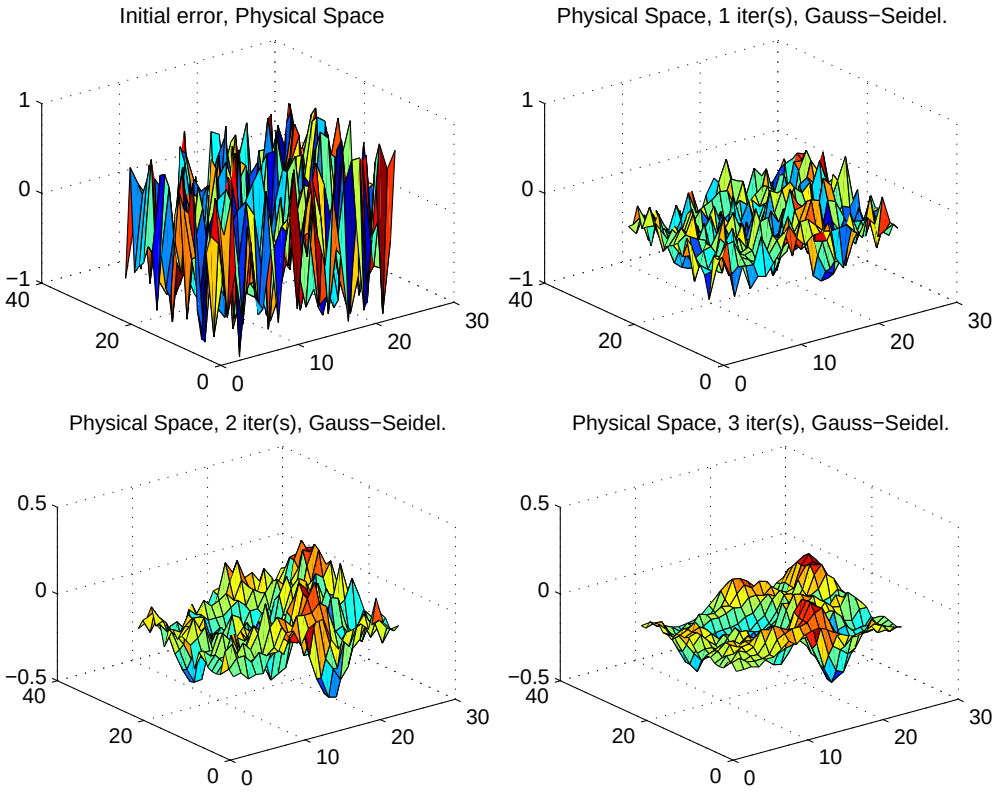

Figure 5: Smoothing by Gauss-Seidel iteration.

where $\alpha > 0$ and $\eta(\nu) \to 0$ as $\nu \to \infty$. This is the case for our model problem where A is the discretization of the minus second derivative (minus Laplacian) and the damped Jacobi iteration, $M = I - \omega h^2 A$, $\omega \in (0, 1/2)$. To show this fact, recall the following lemma [27].

Lemma 1 *Let B be real symmetric semipositive definite matrix such that $0 \leq B \leq I$, and ν is a positive integer. Then*

$$\|B(I - \gamma\, B)^\nu\| \leq \hat{\eta}(\nu), \qquad \hat{\eta}(\nu) = \frac{\nu^\nu}{\gamma\,(\nu+1)^{\nu+1}}$$

where $0 < \gamma \leq 1$.

Proof. Notice that the spectrum $\sigma(B) \in (0, 1)$ and that $\|f(B)\| = \max\{|f(\lambda)|,\, \lambda \in \sigma(B)\}$. Find the maximum of the function $f(x) = x(1 - \gamma\, x)^\nu$. □

Now, we have that there exists a constant C such that $\|h^2\, A\| \leq C$, therefore the matrix $B = \frac{h^2}{C}\, A$ satisfies the conditions of Lemma 1. Hence, the smoothing property is given for $\omega < 1/C$ and with $\alpha = 2$ and $\eta(\nu) = (\frac{1}{\omega})\, \nu^\nu/(\nu+1)^{(\nu+1)}$. For the Gauss-Seidel iteration one can prove that the smoothing property is given with $\alpha = 2$ and $\eta(\nu) \approx 1/\nu$.

2.2 Iterative methods as minimization schemes

The discussion on iterative schemes given above is typical within the classical multilevel framework where multilevel operators are characterized by their properties on Fourier space. In the context of these lecture notes, however, we are interested in the optimization properties of the various multilevel components. In this section we consider iterative schemes from this point of view. Again we use the equivalence between solving the problem $Au = f$ and minimizing the functional

$$J(u) = \frac{1}{2}\, u^T A u - u^T f. \tag{12}$$

Recall that many iterative methods like Jacobi and Gauss-Seidel schemes can be written in terms of a nonsingular matrix Q as follows

$$u^{(\nu+1)} = (I - Q^{-1}A)u^{(\nu)} + Q^{-1}f = u^{(\nu)} + Q^{-1}(f - Au^{(\nu)}) = u^{(\nu)} + Q^{-1}\, r^{(\nu)} \tag{13}$$

where $r^{(\nu)} = f - Au^{(\nu)}$ is the residual for the $u^{(\nu)}$ approximation. With $Q = D/\omega$ we have the damped Jacobi iteration; choosing $Q = D - L$ ($Q = D + U$) the forward (backward) Gauss-Seidel scheme is obtained.

If we use (13) in (12) we have

$$\begin{aligned} J(u^{(\nu+1)}) &= \frac{1}{2}\,(u^{(\nu+1)})^T A u^{(\nu+1)} - (u^{(\nu+1)})^T f \\ &= \frac{1}{2}\,(u^{(\nu)} + Q^{-1}\,r^{(\nu)})^T A (u^{(\nu)} + Q^{-1}\,r^{(\nu)}) - (u^{(\nu)} + Q^{-1}\,r^{(\nu)})^T f \\ &= J(u^{(\nu)}) + \frac{1}{2}\,(Q^{-1}\,r^{(\nu)})^T A\,Q^{-1}\,r^{(\nu)} + (Q^{-1}\,r^{(\nu)})^T (A u^{(\nu)} - f) \\ &= J(u^{(\nu)}) + \frac{1}{2}\,(Q^{-1}\,r^{(\nu)})^T A\,Q^{-1}\,r^{(\nu)} - (Q^{-1}\,r^{(\nu)})^T\,r^{(\nu)}. \end{aligned}$$

We obtain the following

$$J(u^{(\nu+1)}) = J(u^{(\nu)}) - (Q^{-1}\,r^{(\nu)})^T\,[(Q - \frac{1}{2}A)\,Q^{-1}\,r^{(\nu)}].$$

In the case of the Gauss-Seidel iteration we have

$$Q - \frac{1}{2}A = \frac{1}{2}D - \frac{1}{2}(L - U)$$

and therefore

$$(Q^{-1}\,r^{(\nu)})^T\,[(Q - \frac{1}{2}A)\,Q^{-1}\,r^{(\nu)}] = \frac{1}{2}(Q^{-1}\,r^{(\nu)})^T\,[D\,Q^{-1}\,r^{(\nu)}] \geq 0$$

since $(Q^{-1}\,r^{(\nu)})^T\,[(L-U)\,Q^{-1}\,r^{(\nu)}] = 0$ because $L-U$ is antisymmetric. Hence we find that the Gauss-Seidel scheme is a minimizer in the sense that

$$J(u^{(\nu+1)}) \leq J(u^{(\nu)}),$$

where strict inequality holds if $r^{(\nu)} \neq 0$.

Next, consider the case of the damped Jacobi iteration where

$$Q - \frac{1}{2}A = \frac{1}{\omega}(D - \frac{\omega}{2}A).$$

We have the following lemma [52]

Lemma 2 *Let A be real symmetric with $a_{ii} > 0$, and let $\omega > 0$. The matrix $2\,\omega^{-1}D - A$, where $D = diagA$, is positive definite if and only if ω satisfies*

$$0 < \omega \leq \frac{2}{1 - \mu_{min}},$$

where $\mu_{min} \leq 0$ is the minimum eigenvalue of $I - D^{-1}A$.

Proof. Let $B = I - D^{-1}A$. The matrix $2\,\omega^{-1}D - A$ is positive definite if and only if

$$2\,\omega^{-1}I - D^{-1/2}AD^{-1/2} = (2\,\omega^{-1} - 1)I + D^{1/2}\,B\,D^{-1/2} = H$$

is positive definite. The eigenvalues of H are $2\,\omega^{-1} - 1 + \mu_i$ where μ_i are the eigenvalues of B. Since $Tr\,B = 0$ and the μ_i are real, it follows that $\mu_{min} \leq 0$. Therefore H is positive definite if $2\,\omega^{-1} - 1 + \mu_i > 0$. That is if $0 < \omega \leq \frac{2}{1-\mu_{min}} \leq \omega \leq \frac{2}{1-\mu_i}$. □

Therefore $(D - \frac{\omega}{2}A) \geq 0$ for $\omega \in (0, 2/(1 - \mu_{min}))$ and hence

$$J(u^{(\nu+1)}) \leq J(u^{(\nu)}).$$

In a classical multilevel context, the criteria for choosing an iteration scheme is its ability to smooth errors. In an optimization context, we require that the iterative scheme be a minimizer. Thus many other well-known iterative methods can be chosen like, for example, the steepest descent (gradient) method given by

$$Q = Q^{(\nu)} = \frac{{r^{(\nu)}}^T A\, r^{(\nu)}}{{r^{(\nu)}}^T r^{(\nu)}}\, I = \frac{1}{\alpha_\nu} I$$

also notice that $r^{(\nu)} = -J'(u^{(\nu)})$. Therefore we can write

$$u^{(\nu+1)} = u^{(\nu)} + \alpha_\nu r^{(\nu)} = u^{(\nu)} - {Q^{(\nu)}}^{-1}\, J'(u^{(\nu)}).$$

It follows that

$$J(u^{(\nu+1)}) = J(u^{(\nu)}) - \frac{\alpha_\nu}{2}(r^{(\nu)})^T r^{(\nu)}.$$

The iterative schemes discussed above can be interpreted as the process of minimizing the functional J by optimizing successively with respect to each unknown variable (Gauss-Seidel scheme) or in parallel by updating all unknown variables at the same time (Jacobi scheme, steepest descent). In this sense these methods belong to the class of successive or parallel subspace correction (SSC or PSC) methods.

Uniform convergence rates for SC iterations applied to a convex functional $J(u)$ are proven in [48] assuming that J is Gateaux differentiable and that there exist constants K, $L > 0$, $p \geq q > 1$, such that

$$\langle J'(u) - J'(v), w - v \rangle \geq K \|u - v\|_V^p, \tag{14}$$

$$\|J'(u) - J'(v)\|_{V'} \leq L \|u - v\|_V^{q-1}, \tag{15}$$

for all u, $v \in V$, and $\langle \cdot, \cdot \rangle$ is the duality pairing between V and its dual space V'.

2.3 The twolevel scheme and the approximation property

After the application of a few smoothing sweeps, we obtain an approximation $\tilde{u}_h$ whose error $\tilde{e}_h = u_h - \tilde{u}_h$ is smooth. Then $\tilde{e}_h$ can be approximated on a coarser space. We need to express this smooth error as solution of a coarse problem, whose matrix A_H and right-hand side have to be defined. For this purpose notice that in our model problem, A_h is a second-order difference operator and the *residual* $r_h = f_h - A_h \tilde{u}_h$ is a smooth function if $\tilde{e}_h$ is smooth. Obviously, the original equation $A_h u_h = f_h$ and the residual equation $A_h \tilde{e}_h = r_h$ are equivalent. The difference is that $\tilde{e}_h$ and r_h are smooth, therefore we can think of representing them on a coarser grid with mesh size $H = 2h$. We define r_H as the restriction of the fine-grid residual to the coarse grid, that is, $r_H = I_h^H r_h$, where I_h^H is a suitable *restriction* operator (for example, injection). This defines the right-hand side of the coarse problem. Since $\tilde{e}_h$ is the solution of a difference operator which can be represented analogously on the coarse discretization level, we define the following coarse problem

$$A_H \tilde{e}_H = r_H. \tag{16}$$

Here A_H represents the same discrete operator but relative to the grid with mesh size H. Reasonably one expects that $\tilde{e}_H$ be an approximation

to $\tilde{e}_h$ on the coarse grid. Because of its smoothness, we can apply a *prolongation* operator I_H^h to transfer $\tilde{e}_H$ to the fine grid. Therefore, since by definition $u_h = \tilde{u}_h + \tilde{e}_h$, we update the function $\tilde{u}_h$ applying the following *coarse-grid correction* (CGC) step

$$\tilde{u}_h^{new} = \tilde{u}_h + I_H^h \tilde{e}_H.$$

Notice that $\tilde{e}_h$ was a smooth function and the last step has amended $\tilde{u}_h$ by its smooth error. In practice, also the interpolation procedure may introduce HF errors on the fine grid. Therefore it is convenient to complete the twolevel process by applying ν_2 post-smoothing sweeps after the coarse-grid correction.

We summarize the twolevel (TL) procedure with the following algorithm. To emphasize that the iteration $u_h^{(l)} = M u_h^{(l-1)} + N f_h$ is a smoothing procedure, we denote it by $u_h^{(l)} = S_h(u_h^{(l-1)}, f_h)$. When no confusion may arise, we also use S to denote the iteration matrix (in place of) M.

Algorithm 1 (TL scheme)

- *Twolevel method for solving* $A_h u_h = f_h$.
  1. *Pre-smoothing steps on the fine grid:* $u_h^{(l)} = S(u_h^{(l-1)}, f_h)$, $l = 1, \ldots, \nu_1$;
  2. *Computation of the residual:* $r_h = f_h - A_h u_h^{(\nu_1)}$;
  3. *Restriction of the residual:* $r_H = I_h^H r_h$;
  4. *Solution of the coarse-grid problem* $e_H = (A_H)^{-1} r_H$;
  5. *Coarse-grid correction:* $u_h^{(\nu_1+1)} = u^{(\nu_1)} + I_H^h e_H$;
  6. *Post-smoothing steps on the fine grid:* $u_h^{(l)} = S(u_h^{(l-1)}, f_h)$, $l = \nu_1 + 2, \ldots, \nu_1 + \nu_2 + 1$;

A TL scheme starts at the fine level with pre-smoothing, performs a CGC correction solving a coarse-grid auxiliary problem, and ends with

post-smoothing. A pictorial representation of this process where 'fine' is a high level and 'coarse' is a low level looks like a 'V' workflow. This is called V cycle. To solve the problem to a given tolerance, we have to apply the TL V-cycle repeatedly (iteratively). Actually, the TL scheme can be written in the form (1) as stated by the following

Lemma 3 *The iteration matrix of the twolevel scheme is*

$$M_{TL} = S_h^{\nu_2}(I_h - I_H^h(A_H)^{-1}I_h^H A_h)S_h^{\nu_1} \quad , \tag{17}$$

where I_h is the identity and S_h is the smoothing iteration matrix.

For the model problem considered here, it is possible to estimate the spectral radius of M_{TL}. Consider the damped Jacobi smoother with $\omega = 1/2$, assume that I_H^h is the piecewise linear interpolation given by (example)[27]

$$I_H^h = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 1 \end{pmatrix},$$

and I_h^H is restriction by weighting such that $r_H(x_j) = (r_h(x_{j-1}) + 2\,r_h(x_j) + r_h(x_{j+1}))/4$, $j = 2, 4, \ldots, n-1$. In stencil form we have

$$I_H^h = \frac{1}{2}\begin{pmatrix} 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 \end{pmatrix}.$$

With this setting the following theorem is proved [27] using discrete Fourier analysis.

Theorem 2 *Let the TL scheme 1 with $\nu = \nu_1 + \nu_2 \geq 1$. The spectral radius of the iteration matrix M_{TL} given by (17) is bounded by*

$$\rho(M_{TL}) \leq \max\{\chi(1-\chi)^\nu + (1-\chi)\chi^\nu \ : \ 0 \leq \chi \leq 1/2\} =: \rho_\nu < 1 \ ,$$

uniformly with respect to the mesh size h. Hence (17) is a convergent iteration.

In the framework of LMA, a simple and effective way to predict the convergence factor of the TL scheme, for usually moderate values of ν, is to assume that the coarse-grid correction step solves 'exactly' the LF error components, and there is no interaction between high- and low-frequency components. This can be considered an 'ideal' situation. Then the reduction of a suitable norm of the error (e.g., discrete L^2-norm) by one V cycle of the TL method is determined by the reduction of the HF components on the fine grid. For this reason the reduction (convergence) factor, denoted by ρ, can be roughly estimated by

$$\rho_{LMA} = \mu^{\nu_1+\nu_2} \ . \tag{18}$$

A sharper bound can be computed by *twolevel Fourier mode analysis* [17]. For this purpose we need to construct the Fourier symbol of the twolevel coarse-grid correction operator

$$CG_h^H = [I_h - I_H^h \, (A_H)^{-1} \, I_h^H \, A_h].$$

We denote the corresponding symbol by

$$\widehat{CG}_h^H(\theta) = [\hat{I}_h - \hat{I}_H^h(\theta) \, (\widehat{A}_H(2\theta))^{-1} \, \hat{I}_h^H(\theta) \, \widehat{A}_h(\theta)].$$

(Recall that $e^{i\theta x/h} = e^{i(2\theta)x/H}$.) The symbol of the coarse grid operator is

$$\widehat{A}_H(2\theta) = -\frac{2\cos(2\theta) - 2}{H^2}$$

and similarly one constructs $\widehat{A}_h(\theta)$ corresponding to the two harmonics, that is,

$$\widehat{A}_h(\theta) = \begin{bmatrix} -\frac{2\cos(\theta)-2}{h^2} & 0 \\ 0 & \frac{2\cos(\theta)+2}{h^2} \end{bmatrix},$$

The symbol of the restriction operator is (here the hat denotes the Fourier symbol, not the injection operator)

$$\hat{I}_h^H(\theta) = [(1+\cos(\theta))/2 \qquad (1-\cos(\theta))/2],$$

whereas for the injection operator we have $\hat{I}_k^{k-1}(\theta) = 1$. For the linear prolongation operator we have $\hat{I}_{k-1}^k(\theta) = \hat{I}_k^{k-1}(\theta)^T$.

The symbol of the twolevel method is given by

$$\widehat{TG}_h^H(\theta) = \hat{S}_h(\theta)^{\nu_2}\, \widehat{CG}_h^H(\theta)\, \hat{S}_h(\theta)^{\nu_1}.$$

This is an 2×2 matrix corresponding to the two frequency components. In this framework the convergence factor is defined as follows

$$\rho(TG_h^H) = \sup\{r(\widehat{TG}_h^H(\theta)) : \theta \in [-\pi/2, \pi/2)\}.$$

In Table 1, we report estimates of ρ_{QS} as given by (18) and the estimates ρ_{TG} resulting from the twogrid convergence analysis. These are compared with the estimates of ρ_ν by Theorem 2. The estimated ρ_{QS} approximates well the bound ρ_ν provided that $\nu_1+\nu_2$ is small. For large ν, ρ_{QS} has an exponential decay behavior whereas ρ_ν and ρ_{TG} have a slower decay as observed in numerical experiments.

ν	ρ_{QS}	ρ_{TG}	ρ_ν
1	0.5	0.4	0.5
2	0.25	0.19	0.25
3	0.12	0.12	0.12
4	0.06	0.08	0.08

Table 1: Comparison of error reduction factors.

Notice that measuring ρ requires the knowledge of the exact solution. Because this is usually not available, ρ is measured as the asymptotic value of the reduction of a suitable norm (usually the discrete L^2 norm) of the residuals after consecutive TG cycles.

Another theoretical approach to multigrid convergence analysis, related to the smoothing property (11), introduces the *approximation property* to measure how good the coarse-grid solution approximates the fine grid solution. This is expressed by the following estimate

$$\|A_h^{-1} - I_H^h\, A_H^{-1} I_h^H\| \leq c_A\, h^\beta. \tag{19}$$

This estimate actually measures the accuracy between $u_h = A_h^{-1} f_h$ and $I_H^h u_H$ where $u_H = A_H^{-1} I_h^H f_h$. Standard accuracy estimates for our model problem give $\beta = 2$. This is due to the second-order accuracy of the 3-point Laplacian in one dimension and the fact that the error in interpolation and restriction is of second order.

Using the estimates of the smoothing and approximation properties, one can prove convergence of the TG scheme as follows. Consider, for simplicity, $\nu_2 = 0$, i.e. only pre-smoothing is applied. Then for our model problem we have

$$\begin{aligned} \|M_{TG}\| &= \|(I_h - I_H^h (A_H)^{-1} I_h^H A_h) S_h^\nu\| \\ &= \|(A_h^{-1} - I_H^h (A_H)^{-1} I_h^H) A_h S_h^\nu\| \\ &\leq \|A_h^{-1} - I_H^h (A_H)^{-1} I_h^H\| \|A_h S_h^\nu\| \\ &\leq c_A\, \eta(\nu), \end{aligned} \tag{20}$$

where $c_A\, \eta(\nu) < 1$ for sufficiently large ν. Notice that the coarse-grid correction without pre- and post-smoothing is not a convergent iteration, in general. In fact, I_h^H maps from a fine- to a coarse-dimensional space and $I_H^h (A_H)^{-1} I_h^H$ is not invertible. We may have $\rho(I_h - I_H^h (A_H)^{-1} I_h^H A_h) \geq 1$.

We conclude this section showing that the coarse-grid correction step $u_h^{(\nu+1)} = u^{(\nu)} + I_H^h e_H$ with $e_H = (A_H)^{-1} r_H$ and $r_H = I_h^H r_h$, provides an update in the descent direction in the sense that

$$J_h'(u_h^{(\nu)})^T\, (I_H^h\, e_H) < 0,$$

unless $e_H = 0$, occurring at convergence. This means that the TG scheme results in a optimization iteration by choosing a smoothing

scheme with minimization properties and by performing a globalization step along the coarse-grid correction.

We assume that

$$I_h^H = c_I\,(I_H^h)^T \text{ for a constant } c_I > 0.$$

This assumption holds, for example, for I_h^H being full-weighting restriction and I_H^h bilinear interpolation. In this case we have $c_I = (h/H)^d$, with d the space dimension. It follows that

$$\begin{aligned} J_h'(u_h^{(\nu)})^T\,(I_H^h\,e_H) &= -{r_h}^T\,(I_H^h\,e_H) = -\frac{1}{c_I}\,(I_h^H\,r_h)^T\,e_H \\ &= -\frac{1}{c_I}\,r_H^T\,e_H = -\frac{1}{c_I}\,(Ae_H)^T e_H < 0. \end{aligned}$$

Updating along a descent direction is not sufficient to guarantee a reduction of the value of J. For this purpose a line search or an a-priori choice of step-length α is required (globalization) such that

$$J(u^{(\nu)} + \alpha\, I_H^h e_H) < J(u^{(\nu)}).$$

This aspect is discussed later in Sect. 3.2.

2.4 The multilevel scheme

In the TG scheme the size of the coarse problem may be still too large to be solved exactly. Therefore it is convenient to use recursively the TG method to solve (16) thus introducing a further coarse-grid problem. This process can be repeated until a coarsest grid is reached where the corresponding residual equation is inexpensive to solve. This is, roughly speaking, the qualitative description of the multilevel method.

For a more detailed description, let us introduce a sequence of grids with mesh size $h_1 > h_2 > \cdots > h_L > 0$, so that $h_{k-1} = 2\,h_k$. Here $k = 1, 2, \ldots, L$, is called the *level number*. With Ω_{h_k} we denote the set of grid points with grid spacing h_k. The number of interior grid points will be n_k. With V_k we denote the space of functions defined on Ω_{h_k}.

On each level k we define the problem $A_k u_k = f_k$. Here A_k is a $n_k \times n_k$ spd matrix, and u_k and f_k are vectors of size n_k. The transfer among levels is performed by two linear mappings: The restriction I_k^{k-1} and the prolongation I_{k-1}^k operators. With $u_k^{(l)} = S_k(u_k^{(l-1)}, f_k)$ we denote a smoothing iteration.

For variables defined on V_k we introduce the inner product $(\cdot,\cdot)_k$ with associated norm $\|u\|_k = (u,u)_k^{1/2}$. Furthermore, denote with $|u|_k = (A_k u, u)^{1/2}$ the norm induced by A_k.

The following defines the multilevel algorithm.

Algorithm 2 (MG scheme)

- *Multilevel method for solving* $A_k u_k = f_k$.
  1. *If* $k = 1$ *solve* $A_k u_k = f_k$ *exactly.*
  2. *Pre-smoothing steps:* $u_k^{(l)} = S(u_k^{(l-1)}, f_k)$, $l = 1, \ldots, \nu_1$;
  3. *Computation of the residual:* $r_k = f_k - A_k u_k^{(\nu_1)}$;
  4. *Restriction of the residual:* $r_{k-1} = I_k^{k-1} r_k$;
  5. *Set* $u_{k-1} = 0$;
  6. *Call* γ *times the ML scheme to solve* $A_{k-1} u_{k-1} = r_{k-1}$;
  7. *Coarse-grid correction:* $u_k^{(\nu_1+1)} = u_k^{(\nu_1)} + I_{k-1}^k u_{k-1}$;
  8. *Post-smoothing steps:* $u_k^{(l)} = S(u_k^{(l-1)}, f_k)$, $l = \nu_1+2, \ldots, \nu_1+\nu_2+1$;

The multilevel algorithm involves a new parameter (cycle index) γ which is the number of times the MG procedure is applied to the coarse level problem. Since this procedure converges very fast, $\gamma = 1$ or $\gamma = 2$ are the typical values used. For $\gamma = 1$ the multilevel scheme is called V-cycle, whereas $\gamma = 2$ is named W-cycle. It turns out that with a reasonable γ, the coarse problem is solved almost exactly. Therefore in this case the convergence factor of a multilevel cycle equals that of the

corresponding TG method, i.e., approximately $\rho = \mu^{\nu_1+\nu_2}$. Actually in many problems $\gamma = 2$ or even $\gamma = 1$ are sufficient to retain the twolevel convergence. A picture of the multilevel workflow is given in Figure 6. A complete MATLAB linear multilevel code for solving the

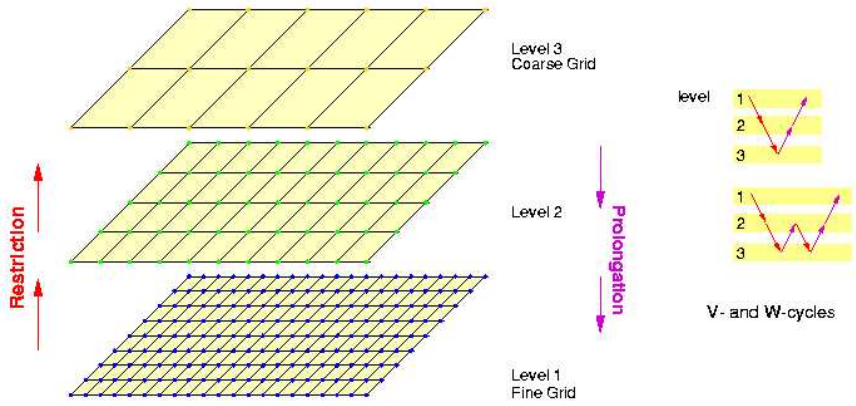

Figure 6: Multilevel setting.

one-dimensional model problem is given in Appendix 6.

Also the multilevel scheme can be expressed in the form (1) as stated by the following lemma

Lemma 4 *The iteration matrix of the multilevel scheme is given in recursive form by the following.*

For $k = 1$ *let* $M_1 = 0$. *For* $k = 2, \ldots, L$:

$$M_k = S_k^{\nu_2}(I_k - I_{k-1}^k(I_{k-1} - M_{k-1}^{\gamma})A_{k-1}^{-1}I_k^{k-1}A_k)S_k^{\nu_1} \qquad (21)$$

where I_k *is the identity,* S_k *is the smoothing iteration matrix, and* M_k *is the multilevel iteration matrix for the level* k.

Proof. To derive (21) consider an initial error $e_k^{(0)}$. The action of ν_1 pre-smoothing steps gives $e_k = S_k^{\nu_1} e_k^{(0)}$ and the corresponding residual $r_k = A_k e_k$. On the coarse grid, this error is given by $e_{k-1} = A_{k-1}^{-1} I_k^{k-1} r_k$. However, in the multilevel algorithm we do not invert A_{k-1} (unless on the coarsest grid) and we apply γ multilevel cycles

instead. That is, denote with $v_{k-1}^{(\gamma)}$ the approximation to e_{k-1} obtained after γ application of M_{k-1}, we have for the error (of the error) $\eta_{k-1}^{(\gamma)} = M_{k-1}^{\gamma}\eta_{k-1}^{(0)}$. That is,

$$e_{k-1} - v_{k-1}^{(\gamma)} = M_{k-1}^{\gamma}(e_{k-1} - v_{k-1}^{(0)}).$$

Following the ML Algorithm 2, we set $v_{k-1}^{(0)} = 0$ (Step 5.). Therefore we have $e_{k-1} - v_{k-1}^{(\gamma)} = M_{k-1}^{(\gamma)}e_{k-1}$ which can be rewritten as $v_{k-1}^{(\gamma)} = (I_{k-1} - M_{k-1}^{\gamma})e_{k-1}$. It follows that

$$\begin{aligned} v_{k-1}^{(\gamma)} &= (I_{k-1} - M_{k-1}^{\gamma})e_{k-1} = (I_{k-1} - M_{k-1}^{\gamma})\, A_{k-1}^{-1} I_k^{k-1}\, r_k \\ &= (I_{k-1} - M_{k-1}^{\gamma})\, A_{k-1}^{-1} I_k^{k-1}\, A_k e_k. \end{aligned}$$

Correspondingly, the coarse-grid correction is $u_k^{(\nu_1+1)} = u_k^{(\nu_1)} + I_{k-1}^{k} v_{k-1}^{(\gamma)}$. In terms of error functions this means that

$$e_k^{(\nu_1+1)} = e_k - I_{k-1}^{k} v_{k-1}^{(\gamma)},$$

substituting the expression for $v_{k-1}^{(\gamma)}$ given above, we obtain

$$e_k^{(\nu_1+1)} = [I_k - I_{k-1}^{k}(I_{k-1} - M_{k-1}^{\gamma})\, A_{k-1}^{-1} I_k^{k-1}\, A_k] e_k^{(\nu_1)}.$$

Finally, consideration of the pre- and post-smoothing sweeps proves the Lemma. □

We now describe a multigrid convergence theory for the Poisson model problem. For this purpose we rewrite the multilevel iteration matrix above in the form of a classical iteration as follows

$$M_k = I_k - B_k A_k.$$

where I_k denotes the identity on V_k. Let $R_k : V_k \to V_k$ be an iteration operator such that $S_k = I_k - R_k\, A_k$ for $k > 1$. (Recall (13) to see that $R_k = Q_k^{-1}$.) Consider

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega, \\ u &= 0 \text{ on } \partial\Omega. \end{aligned} \tag{22}$$

The matrix form of this problem is

$$A_k u_k = f_k \qquad \text{in } V_k. \tag{23}$$

We introduce the following operators. We interpret $I_k^{k-1} : V_k \to V_{k-1}$ as the L_k^2 projection defined by

$$(I_k^{k-1} u, v)_{k-1} = (u, I_{k-1}^k v)_k,$$

for all $u \in V_k$ and $v \in V_{k-1}$. Similarly, let $P_{k-1} : V_k \to V_{k-1}$ be the A_k projection defined by

$$(A_{k-1} P_{k-1} u, v)_{k-1} = (A_k u, I_{k-1}^k v)_k,$$

for all $u \in V_k$ and $v \in V_{k-1}$.

The V-cycle multigrid algorithm to solve (23) in recursive form is given as follows.

Algorithm 3 (MG scheme - recursive form)

- *Set $B_1 = A_1^{-1}$. For $k \geq 2$ define $B_k : V_k \to V_k$ in terms of B_{k-1} as follows. Let $f_k \in V_k$.*

  1. *Define $u^{(l)}$ for $l = 1, \ldots, \nu_1$ by*

  $$u^{(l)} = u^{(l-1)} + R_k(f_k - A_k\, u^{(l-1)}).$$

  2. *Set $u^{(\nu_1+1)} = u^{(\nu_1)} + I_{k-1}^k q$, where*

  $$q = B_{k-1} I_k^{k-1}(f_k - A_k\, u^{(\nu_1)}).$$

  3. *Set $B_k\, f_k = u^{(\nu_1+\nu_2+1)}$, where $u^{(\ell)}$ for $\ell = \nu_1 + 2, \ldots, \nu_1 + \nu_2 + 1$ is given by step 2 (with R_k^T - T means transpose - instead of R_k for a symmetric MG scheme).*

To simplify the analysis of this scheme, we chose $\nu_1 = 1$ and $\nu_2 = 0$, and take $u^{(0)} = 0$. From the definition of P_{k-1}, we see that

$$I_k^{k-1} A_k = A_{k-1} P_{k-1}. \tag{24}$$

Let $S_k\, u = S_k\,(u - u^{(0)}) = u - u^{(1)}$. Now for $u \in V_k$, $k = 2, \ldots, L$, we have

$$\begin{aligned}
(I_k - B_k A_k)\, u &= u - u^{(1)} - I_{k-1}^k q \\
&= S_k\, u - I_{k-1}^k B_{k-1} A_{k-1} P_{k-1} S_k\, u \\
&= [I_k - I_{k-1}^k B_{k-1} A_{k-1} P_{k-1}]\, S_k\, u \\
&= [(I_k - I_{k-1}^k P_{k-1}) + I_{k-1}^k (I_{k-1} - B_{k-1} A_{k-1})\, P_{k-1}] S_k\, u.
\end{aligned} \tag{25}$$

It is immediate to see that this recurrence relation, including post-smoothing, can be written as

$$M_k = S_k\, [(I_k - I_{k-1}^k P_{k-1}) + I_{k-1}^k M_{k-1}\, P_{k-1}] S_k.$$

Clearly, it is equivalent to (21) with $\gamma = 1$. Starting from this recurrence relation, in [12] the following multigrid convergence theorem is proved.

Theorem 3 *Let R_k satisfy (26) and (27) for $k > 1$. Then there exists positive constants $\delta_k < 1$ such that*

$$(A_k\, M_k\, u, u)_k \le \delta_k\, (A_k u, u)_k \qquad \textit{for all } u \in V_k.$$

The two conditions required in Theorem 3 are given below. The first condition concerns the smoothing operator R_k as follows. Let R_k be symmetric and positive definite and S_k be nonnegative. We need $A_k S_k = S_k A_k$. There exists constant $C_R > 0$ and $c > 0$ independent of u and k such that

$$C_R\, \frac{\|u\|_k^2}{\lambda_k} \le (R\, u, u)_k \le c\, (A_k^{-1} u, u)_k \qquad \text{for all } u \in V_k, \tag{26}$$

where λ_k denotes the largest eigenvalue of A_k. In general, notice that if the spectrum $\sigma(S_k) = \sigma(I_k - R_k A_k) \in (-1, 1)$, then there exist positive constants a_0 and a_1 smaller then one such that

$$-a_0\, (A_k u, u)_k \le (A_k (I_k - R_k A_k) u, u)_k \le a_1\, (A_k u, u)_k.$$

This is the same as $(1-a_1)\,(A_k^{-1}u,u)_k \le (Ru,u)_k \le (1+a_0)\,(A_k^{-1}u,u)_k$; see [11]. Compare with the smoothing property (11).

The second assumption is a regularity and approximation assumption. There exists $0<\alpha\le 1$ and a constant $C_\alpha>0$ independent of k such that

$$(A_k(I_k-I_{k-1}^k P_{k-1})\,u,u)_k \le C_\alpha \left(\frac{\|A_k u\|_k^2}{\lambda_k}\right)^{\alpha} (A_k u,u)_k^{1-\alpha} \quad \text{for all } u\in V_k. \tag{27}$$

The case $\alpha=1$ corresponds to full elliptic regularity, $\|u\|_{H^2}\le c\|f\|_{L^2}$. Notice that (27) corresponds to the approximation property (19). In fact, we have

$$I_k - I_{k-1}^k P_{k-1} = \left(A_k^{-1} - I_{k-1}^k\, A_{k-1}^{-1} I_k^{k-1}\right)\, A_k,$$

and if $A_k u_k = f_k$ then $A_{k-1}(P_{k-1}u_k) = I_k^{k-1} f_k$.

In the following, we sketch the proof of Theorem 3. This proof is by induction. For $k=1$ we have $M_1 = I_k - B_1A_1 = I_k - A_1^{-1}A_1 = 0$ and the claim of the theorem is true. Now assume it is true for $k-1$. We have

$$\begin{aligned}
(A_kM_ku,u)_k &= (A_kS_k\,(I_k - I_{k-1}^kP_{k-1})S_ku,u)_k + (A_kS_k\,I_{k-1}^kM_{k-1}\,P_{k-1}S_ku,u)_k\\
&= (A_k\,(I_k - I_{k-1}^kP_{k-1})z,z)_k + (A_k\,I_{k-1}^kM_{k-1}\,P_{k-1}z,z)_k\\
&= (A_k\,(I_k - I_{k-1}^kP_{k-1})z,z)_k + (M_{k-1}\,P_{k-1}z,I_k^{k-1}A_kz)_{k-1}\\
&= (A_k\,(I_k - I_{k-1}^kP_{k-1})z,z)_k + (M_{k-1}\,P_{k-1}z,A_{k-1}P_{k-1}z)_{k-1}\\
&= (A_k\,(I_k - I_{k-1}^kP_{k-1})z,z)_k + (M_{k-1}\,v,A_{k-1}v)_{k-1}\\
&\le (A_k\,(I_k - I_{k-1}^kP_{k-1})z,z)_k + \delta_{k-1}\,(A_{k-1}\,v,v)_{k-1}\\
&= (A_k\,(I_k - I_{k-1}^kP_{k-1})z,z)_k + \delta_{k-1}\,(A_{k-1}P_{k-1}z,P_{k-1}z)_{k-1}\\
&= (A_k\,(I_k - I_{k-1}^kP_{k-1})z,z)_k + \delta_{k-1}\,(A_kz,I_{k-1}^kP_{k-1}z)_k\\
&= (1-\delta_{k-1})\,(A_k\,(I_k - I_{k-1}^kP_{k-1})z,z)_k + \delta_{k-1}\,(A_kz,z)_k
\end{aligned}$$

where we let $z=S_ku$ (the case with ν pre- and post-smoothing sweeps requires $z=S_k^\nu u$) and $v=P_{k-1}z$. To complete the proof of the

theorem, one considers the resulting inequality

$$(A_k M_k u, u)_k \leq (1-\delta_{k-1})\,(A_k\,(I_k - I_{k-1}^k P_{k-1})z, z)_k + \delta_{k-1}\,(A_k z, z)_k. \tag{28}$$

From (27) with $\alpha = 1$ we have

$$(A_k\,(I_k - I_{k-1}^k P_{k-1})z, z)_k \leq C_1\,\frac{\|A_k z\|_k^2}{\lambda_k}.$$

Next, we find a δ independent of k such that Theorem 3 holds for all k. We need the following lemma.

Lemma 5 *The following estimate holds*

$$(A_k\,(I_k - I_{k-1}^k P_{k-1})\,S_k^\nu u, S_k^\nu u)_k \leq \frac{C_1}{2\nu C_R}\,\left(|u|_k^2 - |S_k^\nu u|_k^2\right)$$

Proof.

$$\begin{aligned}
(A_k\,(I_k - I_{k-1}^k P_{k-1})\,S_k^\nu u, S_k^\nu u)_k &\leq C_1\lambda_k^{-1}\,\|A_k S_k^\nu u\|_k^2 \\
&= (C_1/C_R)\,(R_k A_k S_k^\nu u, A_k S_k^\nu u)_k \\
&= (C_1/C_R)\,((I-S_k)S_k^{2\nu} u, A_k u)_k \\
&\leq \frac{C_1}{2\nu C_R}\,\left(|u|_k^2 - |S_k^\nu u|_k^2\right)
\end{aligned}$$

the last inequality follows from

$$((I-S_k)S_k^{2\nu}u, A_k u)_k \leq \frac{1}{2\nu}\sum_{j=0}^{2\nu-1}((I-S_k)S_k^j u, A_k u)_k = \frac{1}{2\nu}\,\left(|u|_k^2 - |S_k^\nu u|_k^2\right)$$

resulting from

$$(1-x)x^{2m} \leq \frac{1}{2m}(1-x)\sum_{j=0}^{2m-1} x^j = \frac{1}{2m}(1-x^{2m}) \text{ for } 0 \leq x \leq 1.$$

□

With this lemma and (28) (with $\delta_{k-1} = \delta$) we obtain

$$(A_k M_k u, u)_k \leq (1-\delta)\,\frac{C_1}{2\nu C_R}\,\left(|u|_k^2 - |S_k^\nu u|_k^2\right) + \delta\,|S_k^\nu u|_k^2.$$

Now, choosing $\delta = C_1/(C_1 + 2\nu C_R)$ we have

$$(A_k M_k u, u)_k \leq \delta\,(A_k u, u)_k$$

where $0 < \delta < 1$ for $\nu \geq 1$ and Theorem 3 is proved with δ independent of k.

Notice that since M_k is symmetric with respect to $(A_k \cdot, \cdot)_k$ it follows that $(A_k M_k^2 u, u)_k \leq \delta^2\,(A_k u, u)_k$. This fact and the additional condition

$$(A_k I_{k-1}^k u, I_{k-1}^k u)_k \leq 2\,(A_{k-1} u, u)_{k-1} \text{ for all } u \in V_k,$$

that characterizes the case of nested spaces, allow to extend the theorem above to the case of W-cycles ($\gamma = 2$) and the same estimate of δ results [11].

3 Multilevel methods for nonlinear problems

Usually, two multilevel approaches to the solution of nonlinear problems are considered. The first is based on a generalization of the ML scheme described above which is called *full approximation storage* (FAS) scheme [14]. The second approach is based on the Newton method and uses the multilevel scheme as inner solver of the linearized equations defining the Jacobian in the Newton step.

An interesting comparison between the FAS and the Newton-ML schemes is presented in [33]. First, it is shown that in terms of computing time, the exact Newton approach is not a viable method. Further, it is demonstrated that the inexact-Newton-ML scheme may provide similar efficiency as the FAS scheme. However, it remains an open issue how many interior ML cycles are possibly needed in the Newton-ML

method to match the FAS efficiency. In this sense the FAS scheme is more robust and we discuss this method in the following.

3.1 The FAS multilevel method

To illustrate the FAS method, consider the discrete nonlinear problem

$$A_k(u_k) = f_k \ , \tag{29}$$

where $A_k(\cdot)$ represents a nonlinear discrete operator on Ω_{h_k}.

The starting point for the FAS scheme is again to define a suitable smoothing process denoted by $u = S(u, f)$. Now suppose to apply a few times this iterative method to (29) obtaining some approximate solution $\tilde{u}_k$. The desired exact correction e_k is defined by $A_k(\tilde{u}_k + e_k) = f_k$. Here the coarse residual equation $A_k e_k = r_k$ makes no sense (nonlinearity, no superposition). Nevertheless the 'correction' equation can instead be written in the form

$$A_k(\tilde{u}_k + e_k) - A_k(\tilde{u}_k) = r_k \ , \tag{30}$$

where $r_k = f_k - A_k(\tilde{u}_k)$. Now assume to represent $\tilde{u}_k + e_k$ on the coarse grid in terms of the coarse-grid variable

$$u_{k-1} := \hat{I}_k^{k-1}\tilde{u}_k + e_{k-1} \ . \tag{31}$$

Since $\hat{I}_k^{k-1}\tilde{u}_k$ and $\tilde{u}_k$ represent the same function but on different grids, the standard choice of the fine-to-coarse linear operator $\hat{I}_k^{k-1}$ is straight injection. The formulation of (30) on the coarse level is obtained by replacing $A_k(\cdot)$ by $A_{k-1}(\cdot)$, $\tilde{u}_k$ by $\hat{I}_k^{k-1}\tilde{u}_k$, and r_k by $I_k^{k-1}r_k = I_k^{k-1}(f_k - A_k(\tilde{u}_k))$, thus we get the FAS equation

$$A_{k-1}(u_{k-1}) = I_k^{k-1}(f_k - A_k(\tilde{u}_k)) + A_{k-1}(\hat{I}_k^{k-1}\tilde{u}_k) \ . \tag{32}$$

This equation can also be written in the form

$$A_{k-1}(u_{k-1}) = I_k^{k-1}f_k + \tau_k^{k-1} \ , \tag{33}$$

where

$$\tau_k^{k-1} = A_{k-1}(\hat{I}_k^{k-1}\tilde{u}_k) - I_k^{k-1}A_k(\tilde{u}_k).$$

Observe that (33) without the τ_k^{k-1} term is the original equation represented on the coarse grid. At convergence $u_{k-1} = \hat{I}_k^{k-1}u_k$, because $f_k - A_k(u_k) = 0$ and $A_{k-1}(u_{k-1}) = A_{k-1}(\hat{I}_k^{k-1}u_k)$. The term τ_k^{k-1} is the fine-to-coarse *defect or residual correction*. That is, the correction to (33) such that its solution coincides with the fine grid solution. This fact allows to reverse the point of view of the multilevel approach [16]. Instead of regarding the coarse level as a device for accelerating convergence on the fine grid, one can view the fine grid as a device for calculating the correction τ_k^{k-1} to the FAS equation. In this way most of the calculation may proceed on coarser spaces.

The direct use of u_{k-1} on fine grids, that is, the direct interpolation of this function by $I_{k-1}^k u_{k-1}$ cannot be used, since it introduces the interpolation errors of the full (possibly oscillatory) solution, instead of the interpolation errors of only the correction e_{k-1}, which is assumed smooth. For this reason the following coarse-grid correction is used

$$u_k = \tilde{u}_k + I_{k-1}^k(u_{k-1} - \hat{I}_k^{k-1}\tilde{u}_k) \ . \tag{34}$$

The complete FAS scheme is summarized below.

Algorithm 4 (FAS scheme)

- *Multilevel FAS method for solving* $A_k(u_k) = f_k$.

  1. *If* $k = 1$ *solve* $A_k(u_k) = f_k$ *exactly.*
  2. *Pre-smoothing steps:* $u_k^{(l)} = S(u_k^{(l-1)}, f_k)$, $l = 1, \ldots, \nu_1$;
  3. *Computation of the residual:* $r_k = f_k - A_k(u_k^{(\nu_1)})$;
  4. *Restriction of the residual:* $r_{k-1} = I_k^{k-1}r_k$;
  5. *Set* $u_{k-1} = \hat{I}_k^{k-1}u_k^{(\nu_1)}$;
  6. *Set* $f_{k-1} = r_{k-1} + A_{k-1}(u_{k-1})$

7. *Call γ times the FAS scheme to solve $A_{k-1}(u_{k-1}) = f_{k-1}$;*
8. *Coarse-grid correction: $u_k^{(\nu_1+1)} = u_k^{(\nu_1)} + I_{k-1}^k(u_{k-1} - \hat{I}_k^{k-1} u_k^{(\nu_1)})$;*
9. *Post-smoothing steps: $u_k^{(l)} = S(u_k^{(l-1)}, f_k)$, $l = \nu_1+2, \ldots, \nu_1+\nu_2+1$;*

The action of one FAS scheme can be also expressed in terms of a (nonlinear) multigrid iteration operator B_k. Starting with an initial approximation $u_k^{(0)}$ the result of one FAS-cycle is then denoted by $u_k = B_k(u_k^{(0)})\, f_k$.

Algorithm 5 (FAS scheme - recursive form)

- *Set $B_1(u_1^{(0)}) \approx A_1^{-1}$ (e.g., iterating with S_1 starting with $u_1^{(0)}$). For $k = 2, \ldots, L$ define B_k in terms of B_{k-1} as follows.*

  1. *Set the starting approximation $u_k^{(0)}$.*
  2. *Pre-smoothing. Define $u_k^{(l)}$ for $l = 1, \ldots, \nu_1$, by*
  $$u_k^{(l)} = S_k(u_k^{(l-1)}, f_k).$$
  3. *Coarse-grid correction. Set*
  $$u_k^{(\nu_1+1)} = u_k^{(\nu_1)} + I_{k-1}^k(u_{k-1} - \hat{I}_k^{k-1} u_k^{(\nu_1)})$$
  where u_{k-1} is given by
  $$u_{k-1} = B_{k-1}(u_{k-1}^{(0)}) \left[I_k^{k-1}(f_k - A_k(u_k^{(\nu_1)})) + A_{k-1}(u_{k-1}^{(0)}) \right].$$
  and $u_{k-1}^{(0)} = \hat{I}_k^{k-1} u_k^{(\nu_1)}$.
  4. *Post-smoothing. Define $u_k^{(l)}$ for $l = \nu_1 + 2, \cdots, \nu_1 + \nu_2 + 1$, by*
  $$u_k^{(l)} = S_k(u_k^{(l-1)}, f_k).$$
  5. *Set $B_k(u_k^{(0)})\, f_k = u_k^{(\nu_1+\nu_2+1)}$.*

A complete FORTRAN FAS multigrid code for solving a two-dimensional Poisson problem is given in Appendix 7. Next, we discuss a multilevel approach to optimization which closely relates with the FAS scheme and provides a framework to investigate convergence of multigrid schemes for nonlinear problems.

3.2 MGOPT: A multilevel optimization scheme

In some recent papers, Nash [40] and Lewis and Nash [35] propose a multilevel approach to optimization problems, called MGOPT, which closely resembles the FAS scheme. One novelty of the MGOPT approach is the extension of the multilevel strategy to optimization problems. This extension exploits the fact that, as we have seen, the coarse-grid correction provides a descent direction. Combining this fact with a line search procedure and a minimizing 'smoothing' iteration, a globally convergent algorithm is obtained. Numerical experiments, e.g. [40], demonstrate that MGOPT greatly improves the efficiency of the underlying optimization scheme used as 'smoother'. However, a coarse-grid correction combined with line search may be necessary for convergence. As shown below, an a-priori choice of the step-length is possible [2].

Consider the following (locally) convex optimization problem

$$\min_{u_k} J_k(u_k) \tag{35}$$

where $k = 1, 2, \ldots, L$, is the resolution or discretization parameter, L denotes the finest resolution, and u_k is the (unconstrained) optimization variable in the space V_k. Among spaces V_k, restriction operators $I_k^{k-1} : V_k \to V_{k-1}$ and prolongation operators $I_{k-1}^k : V_{k-1} \to V_k$ are defined. We require that $(I_k^{k-1}u, v)_{k-1} = (u, I_{k-1}^k v)_k$ for all $u \in V_k$ and $v \in V_{k-1}$, that is, $I_k^{k-1} = c_I\,(I_{k-1}^k)^T$ for a constant $c_I > 0$. Notice that for optimization problems with partial differential equations, the definition of the hierarchy of spaces V_k and of the intergrid transfer operators follows the guidelines of geometrical/algebraical multigrid

techniques. However, in principle the MGOPT framework is not restricted to PDE-based optimization problems and therefore it could be applied to minimization problems without a geometric context. In this case, given J_L and V_L at the finest resolution it is an open issue of how to choose the hierarchy of J_k and of V_k.

We denote with S_k an optimization algorithm (for example the truncated Newton scheme used in [40]) and require that given an initial approximation $u_k^{(0)}$ to the solution of (35), the application of S_k results in sufficient reduction as follows

$$J_k(S_k(u_k^{(0)})) \leq J_k(u_k^{(0)}) - \eta \|\nabla J_k(u_k^{(0)})\|^2 \qquad \text{for some } \eta \in (0,1).$$

The MGOPT scheme is an iterative gradient-based optimization method. Therefore it must be formulated in the same space where the gradient is defined. In the following, we assume a L^2 formulation of the gradient.

To define one cycle of the MGOPT method, it is convenient to consider the minimization problem $\min_{u_k}(J_k(u_k) - (f_k, u_k)_k)$ where $f_L = 0$. Let $u_k^{(0)}$ be the starting approximation at resolution k.

Algorithm 6 (MGOPT scheme)

- *MGOPT method for solving* $\min_{u_k}(J_k(u_k) - (f_k, u_k)_k)$.
  1. *If* $k = 1$ *solve* $\min_{u_k}(J_k(u_k) - (f_k, u_k)_k)$ *exactly, i.e. solve* $\nabla J_k(u_k) = f_k$.
  2. *Pre-optimization. Define* $u_k^{(1)} = S_k(u_k^{(0)})$.
  3. *Setup and solve a coarse-grid minimization problem. Define* $u_{k-1}^{(1)} = I_k^{k-1} u_k^{(1)}$. *Compute the fine-to-coarse gradient correction*

     $$\tau_{k-1} = \nabla J_{k-1}(u_{k-1}^{(1)}) - I_k^{k-1} \nabla J_k(u_k^{(1)}), \qquad f_{k-1} = I_k^{k-1} f_k + \tau_{k-1}.$$

     The coarse-grid minimization problem is given by

     $$\min_{u_{k-1}} \left(J_{k-1}(u_{k-1}) - (f_{k-1}, u_{k-1})_{k-1} \right). \tag{36}$$

Solve (36) *with MGOPT to obtain* u_{k-1}.

4. *Line-search and coarse-grid correction. Perform a line search in the* $I_{k-1}^k(u_{k-1} - I_k^{k-1}u_k^{(1)})$ *direction to obtain* α_k *that minimizes* J_k. *The coarse-grid correction is given by*

$$u_k^{(2)} = u_k^{(1)} + \alpha_k\, I_{k-1}^k(u_{k-1} - I_k^{k-1}u_k^{(1)})$$

5. *Post-optimization. Define* $u_k^{(3)} = S_k(u_k^{(2)})$.

Roughly speaking, the essential guideline for constructing J_k on coarse levels is that it must sufficiently well approximate the convexity properties of the functional at finest resolution. This property and the following remark give an insight into the fact that the coarse-grid correction provides a descending direction; recall the discussion at the end of Sect. 2.3 and see also Lemma 8 below.

Remark 1 With the term $-(f_{k-1}, u_{k-1})_{k-1}$ in Step 3. we have that

$$\nabla\left(J_{k-1}(u_{k-1}) - (f_{k-1}, u_{k-1})_{k-1}\right)|_{u_{k-1}^{(1)}} = I_k^{k-1}\left(\nabla J_k(u_k^{(1)}) - f_k\right).$$

That is, the gradient of the coarse-grid functional at the coarse approximation $u_{k-1}^{(1)} = I_k^{k-1}u_k^{(1)}$ equals the restriction of the gradient of the fine-grid functional at corresponding fine approximation $u_k^{(1)}$.

3.3 Convergence of the MGOPT method

Assume that for each k, J_k is twice Frechét differentiable and $\nabla^2 J_k$ is strictly positive definite and satisfies the condition $(\nabla^2 J_k(u)v, v)_k \geq \beta\|v\|_k^2$ together with $\|\nabla^2 J_k(u) - \nabla^2 J_k(v)\|_k \leq \lambda\,\|u - v\|_k$ uniformly for some positive constants β and λ. We use the expansion

$$J_k(u+z) = J_k(u) + (\nabla J_k(u), z)_k + \frac{1}{2}\int_0^1 (\nabla^2 J_k(u+tz)z, z)_k\, dt. \quad (37)$$

The main tool for our discussion is the following lemma [29].

Lemma 6 For $u, v \in V_k$ assume $(\nabla J_k(u), v)_k \leq 0$ and let γ be such that

$$0 < \gamma \leq -2\delta(\nabla J_k(u), v)_k \left[\int_0^1 (\nabla^2 J_k(u + t\gamma v)\, v, v)_k dt\right]^{-1}$$

for some $\delta \in (0, 1]$. Then

$$-(1-\delta)\gamma(\nabla J_k(u), v)_k \leq J_k(u) - J_k(u+\gamma v) \leq -\gamma(\nabla J_k(u), v)_k. \quad (38)$$

Proof. Set $z = \gamma\, v$ in (37). The first inequality follows from the restriction to γ. The second inequality follows from the positivity of $\nabla^2 J_k$. □

The next lemma provides an explicit estimate for the step-length α_k for the coarse-grid gradient correction in Step 3.

Lemma 7 For $u, v \in V_k$ assume $(\nabla J_k(u), v)_k \leq 0$ and let

$$\alpha(u, v) = \min\{2, \frac{-(\nabla J_k(u), v)_k}{(\nabla^2 J_k(u)v, v)_k + \lambda \|v\|_k^3}\} \quad (39)$$

Then

$$0 \leq -\frac{1}{2}\alpha(u, v)(\nabla J_k(u), v)_k \leq J_k(u) - J_k(u + \alpha(u, v)v). \quad (40)$$

Proof. For the proof it is enough to verify that Lemma 6 may be applied with $\gamma = \alpha(u, v)$ and $\delta = 1/2$. Notice that

$$\int_0^1 (\nabla^2 J_k(u + t\alpha\, v)v, v)_k\, dt \leq (\nabla^2 J_k(u)v, v)_k + \lambda \|v\|_k^3.$$

Therefore we have

$$\alpha(u, v) \leq \frac{-(\nabla J_k(u), v)_k}{(\nabla^2 J_k(u)v, v)_k + \lambda \|v\|_k^3} \leq \frac{-(\nabla J_k(u), v)_k}{\int_0^1 (\nabla^2 J_k(u + t\alpha\, v)v, v)_k dt}.$$

Hence α satisfies the condition of Lemma 6 with $\delta = 1/2$. □

The following lemma states that the MGOPT coarse-grid correction with step-length $0 < \alpha \leq 2$ given by Lemma 7 is a minimizing step. Notice that the above lemmas are formulated for a functional $J_k(u_k)$ and its gradient $\nabla J_k(u_k)$. They hold true considering $J_k(u_k) - (f_k, u_k)_k$ and $\nabla J_k(u_k) - f_k$. Denote with $\hat{J}_k(u_k) = J_k(u_k) - (f_k, u_k)_k$.

Lemma 8 Take $u_k^{(1)} \in V_k$ and define $u_{k-1}^{(1)} = I_k^{k-1} u_k^{(1)} \in V_{k-1}$. Denote with $\hat{J}_{k-1}(u_{k-1}) = J_{k-1}(u_{k-1}) - (f_{k-1}, u_{k-1})_{k-1}$ where $f_{k-1} = I_k^{k-1} f_k + \tau_{k-1}$ and $\tau_{k-1} = \nabla J_{k-1}(u_{k-1}^{(1)}) - I_k^{k-1} \nabla J_k(u_k^{(1)})$. Let $u_{k-1} \in V_{k-1}$ be such that $\hat{J}_{k-1}(u_{k-1}) \leq \hat{J}_{k-1}(u_{k-1}^{(1)})$ and define $q = I_{k-1}^k(u_{k-1} - u_{k-1}^{(1)})$. Then

$$\hat{J}_k(u_k^{(1)} + \alpha(u_k^{(1)}, q)\, q) - \hat{J}_k(u_k^{(1)}) \leq \frac{1}{2}\alpha(u_k^{(1)}, q)(\nabla \hat{J}_k(u_k^{(1)}), q)_k, \quad (41)$$

where $\alpha(u_k^{(1)}, q)$ is defined in Lemma 7 (strict inequality holds if $\hat{J}_{k-1}(u_{k-1}) < \hat{J}_{k-1}(u_{k-1}^{(1)})$).

Proof. The proof follows from Lemma 7 after showing that $(\nabla \hat{J}_k(u_k^{(1)}), q)_k \leq 0$. From (37) we obtain

$$(\nabla \hat{J}_{k-1}(u_{k-1}^{(1)}), u_{k-1} - u_{k-1}^{(1)})_k \leq \hat{J}_{k-1}(u_{k-1}) - \hat{J}_{k-1}(u_{k-1}^{(1)}) \leq 0.$$

Now we have

$$\begin{aligned} (\nabla \hat{J}_k(u_k^{(1)}), q)_k &= (\nabla \hat{J}_k(u_k^{(1)}), I_{k-1}^k(u_{k-1} - u_{k-1}^{(1)}))_k \\ &= (I_k^{k-1}(\nabla \hat{J}_k(u_k^{(1)})), u_{k-1} - u_{k-1}^{(1)})_{k-1} \\ &= (\nabla \hat{J}_{k-1}(u_{k-1}^{(1)}), u_{k-1} - u_{k-1}^{(1)})_{k-1} \leq 0. \quad (42) \end{aligned}$$

For the last equality recall Remark 1 (and the discussion at the end of Sect. 2.3). □

Notice that in Lemma 8 it is not required to solve exactly the coarse minimization problem: find $u \in V_{k-1}$ such that $\hat{J}_{k-1}(u) = \min_{u_{k-1}} \hat{J}_{k-1}(u_{k-1})$. This is (formally) required only on the coarsest grid. The following theorem states convergence of the MGOPT method.

Theorem 4 The MGOPT method described above provides a minimizing iteration and if J is strictly convex then (the index L of the finest level is omitted)

$$\lim_{i \to \infty} \|u^{(i)} - u\| = 0,$$

where $J(u) = \min_v J(v)$ and (i) is the MGOPT cycle index.

Proof. The proof of the first part is by induction. For $k = 2$ we have u where $\hat{J}_1(u) = \min_{u_1} \hat{J}_1(u_1)$ and from Lemma 8 it follows that

$$\begin{aligned} \hat{J}_2(u_2^3) &= \hat{J}_2(S_2(u_2^2)) \le \hat{J}_2(u_2^2) = \hat{J}_2(u_2^1 + \alpha\, I_{k-1}^k(u - I_k^{k-1}u_2^1)) \\ &\le \hat{J}_2(u_2^1) = \hat{J}_2(S_2(u_2^0)) \le \hat{J}_2(u_2^0). \end{aligned} \tag{43}$$

If $\hat{J}_2(u_2^0) > \min_{u_2} \hat{J}_2(u_2)$, then (43) holds with strict inequality.

For $k > 2$, due to the induction hypothesis and because of Lemma 8 the theorem holds.

The sequence $\{u_L^{(i)}\}_{i\ge1}$ is in the compact set $A = \{v \in V_L : J_L(v) \le J_L(u_L^0)\}$ and $\{J_L(u_L^{(i)})\}_{i\ge1}$ is a non-increasing sequence in the compact set $V = \{J_L(v) : v \in A\}$, so this sequence converges. We can write $\lim_{i\to\infty}(J_L(u_L^{(i)}) - J_L(u_L)) = 0$. Strict convexity and (37) give that $\lim_{i\to\infty} \|u_L^{(i)} - u_L\|_L = 0$. □

Linear multigrid schemes including algebraic multilevel methods [6, 41, 43] can be interpreted as MGOPT schemes for the quadratic functional

$$J_k(u) = \frac{1}{2}(u, A_k u)_k - (u, b_k)_k, \quad u \in V_k,$$

where $V_k = \mathbb{R}^{n_k}$ and A_k is a $n_k \times n_k$ symmetric positive definite matrix. Consider $n_{k-1} < n_k$ and take I_{k-1}^k be full rank. Then, with the Galerkin formula $A_{k-1} = I_k^{k-1} A_k I_{k-1}^k$ and $b_{k-1} = I_k^{k-1} b_k$, one obtains a suitable coarse functional $J_{k-1}(u) = \frac{1}{2}(u, A_{k-1}u)_{k-1} - (u, b_{k-1})_{k-1}$. A twolevel analysis of the MGOPT scheme applied to this problem reveals that we have convergence for $\alpha = 1$. In fact, consider (39) with $v = I_H^h e_H$, we have

$$-\frac{(\nabla J_h, I_H^h e_H)_h}{(A_h I_H^h e_H, I_H^h e_H)_h} = \frac{(I_h^H r_h, e_H)_H}{(A_h I_H^h e_H, I_H^h e_H)_h} = \frac{(A_H e_H, e_H)_H}{(I_h^H A_h I_H^h e_H, e_H)_H} = 1$$

(Notice that in the linear case $\lambda = 0$.)

The MGOPT convergence theory given above applies also to analyze the FAS scheme. For this purpose we assume the existence of a functional J_k such that $\nabla J_k(u_k) = A_k(u_k) - f_k$. Then, subject to the

conditions given above, convergence of the FAS scheme is proved if one proves that α given by (39) is always greater or equal to one.

3.4 The full multilevel method

When dealing with nonlinear problems it may be essential to start the iterative procedure from a good initial approximation. The multilevel setting suggests a natural way of how to get this approximation. Suppose to start the solution process from a coarse working level $K < M$ where the discretized problem $A_\ell(u_\ell) = f_\ell$ with $\ell = K$ is easily solved. The idea is to interpolate this solution to the next finer working level as initial approximation for the iterative process to solve $A_{\ell+1}(u_{\ell+1}) = f_{\ell+1}$ as follows

$$u_{\ell+1} = \tilde{I}_\ell^{\ell+1} u_\ell. \tag{44}$$

Thereafter the FAS (or ML) solution process at level $\ell + 1$ is applied. The idea of using a coarse-grid approximation as a first guess for the solution process on a finer grid is known as *nested iteration*. The algorithm obtained by combining the multilevel scheme with nested iteration is called *full multilevel* (FML) method; see Figure 7. The interpolation operator (44) used in the FML scheme is called FML interpolator. Because of the improvement on the initial solution at each starting level, the FML scheme results to be more efficient than the iterative application of the multilevel cycle without FML initialization.

Algorithm 7 (FML scheme)

- *FML method for solving* $A_L(u_L) = f_L$.
  1. *For* $\ell = K < L$ *set initial approximation* u_ℓ*;*
  2. *If* $\ell < L$ *then interpolate to the next finer working level:* $\tilde{u}_{\ell+1} = \tilde{I}_\ell^{\ell+1} u_\ell$*;*
  3. *Apply FAS (or ML) scheme to solve* $A_{\ell+1}(u_{\ell+1}) = f_{\ell+1}$*, starting with* $\tilde{u}_{\ell+1}$*;*

4. Set $\ell := \ell + 1$. If $\ell < L$ go to 2; else stop.

On each current working level one applies N-FAS (or ML) cycles and then the algorithm is called N-FML scheme.

Within the N-FML algorithm an estimate of the degree of accuracy can be obtained by comparison of solutions at different levels. Denote with u_ℓ the solution on the level ℓ after N-FAS (or ML) cycles. Then in the FML method this solution is interpolated to level $\ell + 1$ to serve as a first approximation for this working level. At the end of N cycles on this level, one obtains $u_{\ell+1}$. An estimate of the (maximum) norm of the solution error on level ℓ can be defined as

$$E_\ell = \max_{\Omega_{h_\ell}} |u_\ell - \hat{I}^\ell_{\ell+1} u_{\ell+1}| \ , \tag{45}$$

and so on finer levels.

To show the efficiency of the full multilevel approach we consider a three dimensional Poisson problem with Dirichlet boundary conditions:

$$\begin{cases} -(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}) = 3\sin(x+y+z) & \text{in} \quad \Omega = (0,2)^3 \\ u(x,y,z) = \sin(x+y+z) & \text{for} \quad (x,y,z) \in \partial\Omega \end{cases} \tag{46}$$

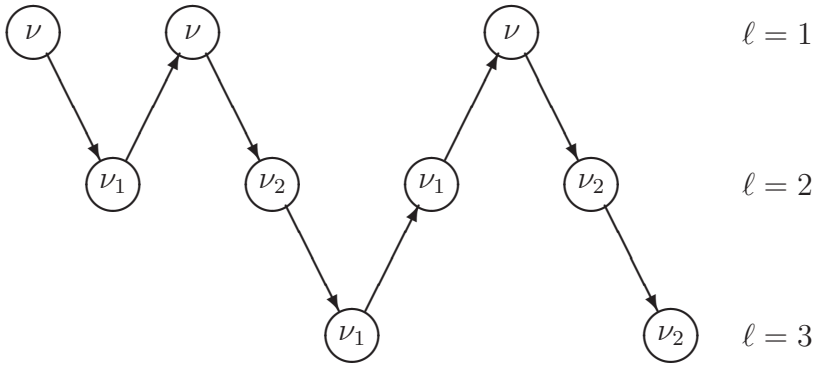

Figure 7: The FML scheme.

Finite difference approximations to (46) are obtained in much the same way as was done in the one-space variable case. If on each level ℓ, we choose a uniform mesh with grid size $h_\ell = \frac{2}{n_\ell+1}$, $n_\ell = 2^\ell - 1$, a direct application of the discretization scheme used in the 1D case gives

$$-\left(\frac{u^\ell_{i+1jk} - 2u^\ell_{ijk} + u^\ell_{i-1jk}}{h^2_\ell} + \frac{u^\ell_{ij+1k} - 2u^\ell_{ijk} + u^\ell_{ij-1k}}{h^2_\ell} + \frac{u^\ell_{ijk+1} - 2u^\ell_{ijk} + u^\ell_{ijk-1}}{h^2_\ell}\right)$$
$$= 3\sin(ih_\ell, jh_\ell, kh_\ell) \ , \quad i,j,k = 1,\ldots,n_\ell \ . \tag{47}$$

This is the 7-point stencil approximation which is $O(h^2)$ accurate. To solve this problem we employ the FAS scheme with $\gamma = 1$ and $L = 7$ with Gauss-Seidel smoothing, $\nu_1 = 2$ and $\nu_2 = 1$. Its smoothing factor estimated by local mode analysis is $\mu = 0.567$. Hence, by this analysis, the expected reduction factor is $\rho^* = \mu^{\nu_1+\nu_2} = 0.18$. The observed reduction factor is ≈ 0.20.

Here $\hat{I}^{k-1}_k$ is simple injection and I^{k-1}_k and I^k_{k-1} are the full weighting and (tri-)linear interpolation, respectively. Finally, the FML operator $\tilde{I}^{\ell+1}_\ell$ is cubic interpolation [16, 27]. Now let us discuss the optimality of the FML algorithm constructed with these components.

To show that the FML scheme is able to solve the given discrete problem at a minimal cost, we compute E_ℓ and show that it behaves like h^2_ℓ, demonstrating convergence. This means that the ratio $E_\ell/E_{\ell+1}$ at convergence should be a factor $h^2_\ell/h^2_{\ell+1} = 4$. Results are reported in Table 2. In the 10-FML column of we actually observe the h^2 behavior which can also be seen with smaller $N = 3$ and $N = 1$ FML schemes. In fact, as reported in Table 2, the 1-FML scheme gives the same order of magnitude of errors as the 10-FML scheme. Therefore the choice $N = 1$ in a FML cycle is suitable to solve the problem to second-order accuracy.

To estimate the amount of work invested in the FML method, let us define the work unit (WU) [14], i.e. the computational work of one smoothing sweep on the finest level M. The number of corresponding arithmetic operations is linearly proportional to the number of grid

level	10-FML	3-FML	1-FML
3	$6.75\,10^{-4}$	$6.79\ \ 10^{-4}$	9.44×10^{-4}
4	$1.73\,10^{-4}$	$1.75\ \ 10^{-4}$	2.34×10^{-4}
5	$4.36\,10^{-5}$	$4.40\ \ 10^{-5}$	5.92×10^{-5}
6	$1.09\,10^{-5}$	$1.10\ \ 10^{-5}$	1.48×10^{-5}

Table 2: The estimated solution error for various N-FML cycles.

points. On the level $\ell \leq M$ the work involved is $(\frac{1}{2})^{3(M-\ell)}WU$, where the factor $\frac{1}{2}$ is given by the mesh size ratio $h_{\ell+1}/h_\ell$ and the exponent 3 is the number of spatial dimensions. Thus a multilevel cycle that uses $\nu = \nu_1 + \nu_2$ relaxation sweeps on each level requires

$$W_{cycle} = \nu \sum_{k=1}^{L} (\frac{1}{2})^{3(L-k)} WU < \frac{8}{7}\nu WU \ \ ,$$

ignoring transfer operations. Hence the computational work employed in a N-FML method is roughly

$$W_{FMG} = N \sum_{\ell=2}^{L} (\frac{1}{2})^{3(L-\ell)} W_{cycle} \ \ ,$$

ignoring the FML interpolation and work on the coarsest grid. This means that, using the 1-FML method, we solve the discrete 3D Poisson problem to second-order accuracy with a number of computer operations which is proportional to the number of unknowns on the finest grid. In the present case we have FML Work of $\approx 4WU$.

4 Multilevel schemes for optimality systems

In this section we discuss the discretization and the multigrid solution of optimal control problems. These consist of a dynamical or equilibrium system, a description of the control mechanism, and a criterion defining the cost functional, that models the purpose of the control and describes the cost of its action. An optimal control problem is then formulated as the minimization of the cost functional where the state of the system is characterized by the modeling equations and the action of the control. This is a constrained minimization problem. The necessary conditions for such a minimum result in a set of coupled equations called the optimality system.

Further details on the multigrid solution of steady and unsteady optimal control problems can be found in [3]–[10].

4.1 Optimality systems

Consider the optimal control problem

$$\begin{cases} \min_{u\in U} J(y,u), \\ \qquad e(y,u) \;=\; 0 \quad \text{in } \Omega, \end{cases} \tag{48}$$

where y and u denote the state- and control variables of a controlled partial differential equation expressed as $e(y,u)=0$, with $e : Y \times U \rightarrow Z$ for appropriate Hilbert spaces Y, U, and Z. Ω is an open bounded set in $\mathbf{R}^d$. The cost functional J is formally given by

$$J(y,u) = h(y) + \nu\, g(u), \tag{49}$$

where $\nu > 0$ is the weight of the cost of the control. Here g and h are required to be continuously differentiable, bounded from below, and such that $g(u) \rightarrow \infty$ as $||u|| \rightarrow \infty$. Allowing g and h to be locally non-convex and e to be possibly nonlinear, (48) may have multiple extremals including minima, maxima, and saddle points.

Local minima satisfy the first-order necessary conditions. To define these conditions consider the Lagrangian

$$L(y,u,p) = J(y,u) + \langle e(y,u), p\rangle_{Z,Z^*},$$

where p is the Lagrange multiplier, the adjoint variable. By equating to zero the Frechét derivatives of L with respect to the triple (y,u,p), we obtain the following optimality system

$$\begin{aligned} e(y,u) &= 0, \\ e_y(y,u)^*\, p &= -h'(y), \\ \nu\, g'(u) + e_u^*\, p &= 0. \end{aligned} \tag{50}$$

Numerical approximations to solutions of (50) can be obtained, after discretization, using the multilevel schemes discussed below. In the following we shall consider optimal control problems where solutions to (50) are local minima. More general situations where these solutions are not necessarily local minima but rather only extremal points are discussed in Section 5.

4.2 Elliptic optimal control problems

We discuss an optimal control problem governed by a linear elliptic equation. Consider a model problem representing a material plate defining a two-dimensional convex domain Ω. For the state y of the material we choose the temperature distribution which is maintained equal to zero along the boundary. This system is governed by the following equation

$$\begin{cases} \Delta y &= f \quad \text{in } \Omega, \\ y &= 0 \quad \text{on } \partial\Omega. \end{cases} \tag{51}$$

The setting above suggests that we may control the temperature distribution y to come close to a given target profile $z \in L^2(\Omega)$ by acting with an additional distributed source term u, the control function.

The corresponding optimal control problem is formulated as follows

$$\begin{cases} \min_{u \in U_{ad}} J(y,u) & & & \\ \qquad\qquad \Delta y &= u + f & \text{in } \Omega, \\ \qquad\qquad y &= 0 & \text{on } \partial\Omega, \end{cases} \tag{52}$$

where we assume that $u \in U_{ad} = L^2(\Omega)$ being the set of admissible controls.

The cost functional J is of the tracking type and is given by

$$J(y,u) = \frac{1}{2}\|y - z\|^2_{L^2(\Omega)} + \frac{\nu}{2}\|u\|^2_{L^2(\Omega)}, \tag{53}$$

where $\nu > 0$ is the weight of the cost of the control.

Existence of solutions to (52)–(53) can be easily established [36].

Optimal solutions are characterized by the following optimality system

$$\begin{aligned} \Delta y &= u + f && \text{in } \Omega, \\ y &= 0 && \text{on } \partial\Omega, \\ \Delta p &= -(y - z) && \text{in } \Omega, \\ p &= 0 && \text{on } \partial\Omega, \\ \nu u - p &= 0 && \text{in } \Omega. \end{aligned} \tag{54}$$

We refer to the first differential equation of (54) as the state equation and to the second one as the adjoint equation. The last equation in (54) gives the optimality condition.

4.3 Finite difference discretization

In this section we discuss finite differences discretization of the optimality system given above. Consider a sequence of grids $\{\Omega_h\}_{h>0}$ given by

$$\Omega_h = \{\mathbf{x} \in \mathbf{R}^2 : x_i = s_i\, h, \quad s_i \in \mathbb{Z}\} \cap \Omega.$$

We assume that Ω is a rectangular domain and that the values of the mesh size h are chosen such that the boundaries of Ω coincide with

grid lines. For grid functions v_h and w_h defined on Ω_h we introduce the discrete L^2-scalar product

$$(v_h, w_h)_{L_h^2} = h^2 \sum_{\mathbf{x}\in\Omega_h} v_h(\mathbf{x})\, w_h(\mathbf{x}),$$

with associated norm $|v_h|_0 = (v_h, v_h)_{L_h^2}^{1/2}$. We also need $|v_h|_\infty = \max_{\mathbf{x}\in\Omega_h} |v_h(\mathbf{x})|$.

First-order backward and forward partial derivatives of v_h in the x_i direction are denoted by ∂_i^- and ∂_i^+, respectively, and given by

$$\partial_i^- v_h(\mathbf{x}) = \frac{v_h(\mathbf{x}) - v_h(\mathbf{x} - \hat{i}\, h)}{h} \text{ and } \partial_i^+ v_h(\mathbf{x}) = \frac{v_h(\mathbf{x} + \hat{i}\, h) - v_h(\mathbf{x})}{h}$$

where $\hat{i}$ denotes the i coordinate direction vector and v_h is extended by 0 on grid points outside of Ω; see [28]. In this framework, the discrete H^1-product is given by

$$|v_h|_1 = \left(|v_h|_0^2 + \sum_{i=1}^{2} |\partial_i^- v_h|_0^2 \right)^{1/2}.$$

The spaces L_h^2 and H_h^1 consist of the sets of grid functions v_h endowed with $|v_h|_0$, respectively $|v_h|_1$, as norm. We need the following lemma [45].

Lemma 9 (Poincaré–Friedrichs inequality for finite differences) For any grid function v_h, there exists a constant c_*, independent of v_h and h, such that

$$|v_h|_0^2 \leq c_* \sum_{i=1}^{2} |\partial_i^- v_h|_0^2. \tag{55}$$

(Note: for $\Omega = (0,1) \times (0,1)$, $c_* = 1/4$.)

Functions in $L^2(\Omega)$ and $H^1(\Omega)$ are approximated by grid functions defined through their mean values with respect to elementary cells $[x_1 - \frac{h}{2}, x_1 + \frac{h}{2}] \times [x_2 - \frac{h}{2}, x_2 + \frac{h}{2}]$; see [28] for more details. For sufficiently

smooth functions $v \in C^k(\bar{\Omega})$ (resp. $f \in C^k(\Omega)$), $k = 0, 1, \ldots,$ we denote with $(R_h v)(x) = v(x)$ (resp. $(\tilde{R}_h f)(x) = f(x)$) the restriction operator on $\bar{\Omega}_h$ (resp. Ω_h).

The (standard) second-order five-point approximation to the Laplacian with homogeneous Dirichlet boundary conditions is defined by

$$\Delta_h = \partial_1^+ \partial_1^- + \partial_2^+ \partial_2^-.$$

We have the following consistency result

$$|\Delta_h R_h v - \tilde{R}_h \Delta v|_\infty \leq c\, h^2 \|v\|_{C^4(\bar{\Omega})}; \tag{56}$$

see, e.g., [28].

After discretization and elimination of the variable u_h we have the following discrete optimality system

$$\Delta_h y_h - p_h/\nu = \tilde{f}_h, \tag{57}$$

$$\Delta_h p_h + y_h = \tilde{z}_h, \tag{58}$$

where $\tilde{f}_h = \tilde{R}_h f$ and $\tilde{z}_h = \tilde{R}_h z$.

Now consider the inner product of (57) by $\nu\, y_h$ and of (58) by p_h and take the sum of the two resulting equations. We obtain

$$\nu\, (\Delta_h y_h, y_h)_{L_h^2} + (\Delta_h p_h, p_h)_{L_h^2} = \nu\, (\tilde{f}_h, y_h)_{L_h^2} + (\tilde{z}_h, p_h)_{L_h^2},$$

which implies that

$$\nu\, (-\Delta_h y_h, y_h)_{L_h^2} + (-\Delta_h p_h, p_h)_{L_h^2} \leq \nu\, |(\tilde{f}_h, y_h)_{L_h^2}| + |(\tilde{z}_h, p_h)_{L_h^2}|.$$

Because $(-\Delta_h v_h, v_h)_{L_h^2} = \sum_{i=1}^2 |\partial_i^- v_h|_0^2$ and using Lemma 9, we obtain

$$\nu\, |y_h|_0^2 + |p_h|_0^2 \leq c_*\, \nu\, |(\tilde{f}_h, y_h)_{L_h^2}| + c_*\, |(\tilde{z}_h, p_h)_{L_h^2}|.$$

Applying the Cauchy-Schwarz and Cauchy inequalities on the right-hand side of this expression results in

$$\nu\, |y_h|_0^2 + |p_h|_0^2 \leq c\, (\nu\, |\tilde{f}_h|_0^2 + \, |\tilde{z}_h|_0^2), \tag{59}$$

where $c = c_*/(2 - c_*)$.

Using (59), we are now able to determine the degree of accuracy of the optimal solution. For this purpose, notice that (57)–(58) hold true with y_h and p_h replaced by their respective error functions, and with $\tilde{f}_h$ and $\tilde{z}_h$ replaced by the truncation error for Δ_h estimated by (56). Further notice that dividing (59) by ν and recalling that $u_h = p_h/\nu$, we obtain the estimate for the control from $|y_h|_0^2 + \nu\,|u_h|_0^2 \leq c\,(|\tilde{f}_h|_0^2 + |\tilde{z}_h|_0^2/\nu)$. These statements are summarized in the following theorem.

Theorem 5 *Let $y \in C^4(\bar{\Omega})$, and $p \in C^4(\bar{\Omega})$, be solutions to (54), and let y_h and p_h be solutions to (57)–(58). Then there exists a constant c, depending on Ω, and independent of h, such that*

$$|y_h - R_h y|_0^2 + \frac{1}{\nu}\,|p_h - R_h p|_0^2 \leq c\,(h^4\,\|y\|^2_{C^4(\bar{\Omega})} + h^4\,\frac{1}{\nu}\,\|p\|^2_{C^4(\bar{\Omega})}).$$

That means that the numerical solution is second-order accurate.

4.4 Smoothing iteration

Let $\mathbf{x} \in \Omega_h$, where $\mathbf{x} = (ih, jh)$ and i, j index the grid points, e.g., lexicographically. Denote with ω_{ij} the set of grid index pairs s, t of the stencil of Δ_h centered at i, j. That is, for Δ_h centered at $0, 0$ we have $\omega_{ij} = \{0, 0; 1, 0; -1, 0; 0, 1; 0, -1\}$. Correspondingly, denote with c_{st}, $s, t \in \omega_{ij}$, the s, t coefficient of the stencil (multiplied by h^2) centered at i, j. In the example we have $c_{00} = -4$, $c_{10} = 1$, etc.. Using this notation, we can express the action of Δ_h on the function v_h in the following compact form

$$\Delta_h v_h|_{ij} = \frac{1}{h^2}\,(\sum_{s,t\in\omega_{ij},\, s,t\neq i,j} c_{st}\,v_{st} - c_{ij}v_{ij}).$$

Consider (69) and (70) at $\mathbf{x}$, we have

$$\sum_{s,t\in\omega_{ij},\, s,t\neq i,j} c^y_{st}\, y_{st} - c^y_{ij}\, y_{ij} - h^2\, u_{ij} \;=\; h^2 \tilde{f}_{ij}, \tag{60}$$

$$\sum_{s,t\in\omega_{ij},\, s,t\neq i,j} c^p_{st}\, p_{st} - c^p_{ij}\, p_{ij} + h^2\, y_{ij} \;=\; h^2\, \tilde{z}_{ij}, \tag{61}$$

$$\nu u_{ij} - p_{ij} \;=\; 0, \tag{62}$$

where $\tilde{f}$ and $\tilde{z}$ represent f_h and z_h including the defect corrections. Notice that for convenience we distinguish between the coefficient c_{st} for the y and p variables by writing c^y_{st} and c^p_{st}, respectively.

To ease notation we set

$$A_{ij} = \sum_{s,t\in\omega_{ij},\, s,t\neq i,j} c^y_{st}\, y_{st} - h^2 \tilde{f}_{ij} \quad \text{and} \quad B_{ij} = \sum_{s,t\in\omega_{ij},\, s,t\neq i,j} c^p_{st}\, p_{st} - h^2 \tilde{z}_{ij}.$$

Here A_{ij} and B_{ij} are considered constant during the update of the variables at i, j. Summarizing, we have the following system for the three scalar variables y_{ij}, p_{ij}, and u_{ij}:

$$A_{ij} - c^y_{ij}\, y_{ij} - h^2\, u_{ij} \;=\; 0, \tag{63}$$

$$B_{ij} - c^p_{ij}\, p_{ij} + h^2\, y_{ij} \;=\; 0, \tag{64}$$

$$\nu u_{ij} - p_{ij} \;=\; 0. \tag{65}$$

This system can be solved immediately and its solution gives an update to y_{ij}, p_{ij}, and u_{ij} defining a collective (all-at-once) Gauss-Seidel step.

In fact, we obtain y_{ij} and p_{ij} as functions of u_{ij} as follows

$$y_{ij} = (A_{ij} - h^2\, u_{ij})/c^y_{ij}, \tag{66}$$

and

$$p_{ij} = (c^y_{ij}\, B_{ij} + h^2\, A_{ij} - h^4 u_{ij})/c^y_{ij}\, c^p_{ij}. \tag{67}$$

Now to obtain the u_{ij} update, replace the expression for p_{ij} in the optimality condition. We obtain

$$u_{ij} = \frac{1}{\nu + h^4/c^y_{ij}\, c^p_{ij}}\, (c^y_{ij}\, B_{ij} + h^2\, A_{ij})/c^y_{ij}\, c^p_{ij}. \tag{68}$$

With the new value of u_{ij} given, new values for y_{ij} and p_{ij} are obtained from (66) and (67), respectively. Clearly, also in this case we can remove the variable u_{ij} altogether.

4.5 A FAS multilevel scheme

We complete the description of the multilevel solution process in this section illustrating the FAS structure. We have the following system

$$\begin{aligned} \Delta_k y_k - p_k/\nu &= f_k, \quad (69)\\ \Delta_k p_k + y_k &= z_k, \quad (70) \end{aligned}$$

Consider a current approximation to the solution of this system denoted by $\mathbf{w}_k = (y_k, p_k, u_k)$ and apply ν_1-times the iterative scheme described in the previous section, so that the main components of the error that are left are smooth. Then we start a coarse-grid correction procedure to solve for these components. First, a coarse level problem is constructed on the grid with mesh size h_{k-1} given by

$$\begin{aligned} \Delta_{k-1} y_{k-1} - p_{k-1}/\nu &= I_k^{k-1} f_k + \tau(y)_k^{k-1}, \quad (71)\\ \Delta_{k-1} p_{k-1} + y_{k-1} &= I_k^{k-1} z_k + \tau(p)_k^{k-1}, \quad (72) \end{aligned}$$

where $\tau(y)_k^{k-1}$ and $\tau(p)_k^{k-1}$ are fine-to-coarse defect corrections defined by

$$\begin{aligned} \tau(y)_k^{k-1} &= \Delta_{k-1}\hat{I}_k^{k-1} y_k - \hat{I}_k^{k-1} p_k/\nu - I_k^{k-1}(\Delta_k y_k - p_k/\nu), \quad (73)\\ \tau(p)_k^{k-1} &= \Delta_{k-1}\hat{I}_k^{k-1} p_k + \hat{I}_k^{k-1} y_k - I_k^{k-1}(\Delta_k p_k + y_k). \quad (74) \end{aligned}$$

Here $\hat{I}_k^{k-1}$ is straight injection and I_{k-1}^k, I_k^{k-1} are as follows

$$I_{k-1}^k = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \text{ and } I_k^{k-1} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \quad (75)$$

Once the coarse level problem is solved, which gives $\mathbf{w}_{k-1} = (y_{k-1}, p_{k-1}, u_{k-1})$, the coarse-grid correction follows

$$y_k^{new} = y_k + I_{k-1}^k (y_{k-1} - \hat{I}_k^{k-1} y_k), \tag{76}$$
$$p_k^{new} = p_k + I_{k-1}^k (p_{k-1} - \hat{I}_k^{k-1} p_k). \tag{77}$$

This is followed by ν_2 post-smoothing steps.

A complete MATLAB FAS multilevel code for solving (69)-(70) including results of numerical experiments is given in Appendix 8.

4.6 Local Fourier convergence analysis

We now proceed with a local Fourier analysis [17, 49, 51] to investigate the convergence properties of the twolevel method applied to the optimality system above.

First, let us recall the local Fourier setting. Consider a sequence of (infinite) grids, $\Gamma_k = \{(ih_k, jh_k), i, j \in \mathbb{Z}\}$. On these grids we define the Fourier components:

$$\phi_k(\boldsymbol{\theta}, \mathbf{x}) = e^{i\theta_1 x_1/h_k} e^{i\theta_2 x_2/h_k}.$$

For any low frequency $\boldsymbol{\theta} = (\theta_1, \theta_2) \in [-\pi/2, \pi/2)^2$, we consider

$$\boldsymbol{\theta}^{(0,0)} := (\theta_1, \theta_2), \qquad \boldsymbol{\theta}^{(1,1)} := (\overline{\theta_1}, \overline{\theta_2}),$$
$$\boldsymbol{\theta}^{(1,0)} := (\overline{\theta_1}, \theta_2), \qquad \boldsymbol{\theta}^{(0,1)} := (\theta_1, \overline{\theta_2}),$$

where

$$\overline{\theta_i} = \begin{cases} \theta_i + \pi \text{ if } \theta_i < 0, \\ \theta_i - \pi \text{ if } \theta_i \geq 0. \end{cases}$$

We have $\phi(\boldsymbol{\theta}^{(0,0)}, \mathbf{x})_k = \phi(\boldsymbol{\theta}^{(1,1)}, \mathbf{x})_k = \phi(\boldsymbol{\theta}^{(1,0)}, \mathbf{x})_k = \phi(\boldsymbol{\theta}^{(0,1)}, \mathbf{x})_k$ for $\boldsymbol{\theta}^{(0,0)} \in [-\pi/2, \pi/2)^2$ and $\mathbf{x} = (x_1, x_2) \in \Gamma_{k-1}$. That is, we have a quadruples of distinct Fourier components that coincide (aliases) on Γ_{k-1}.

Denote with $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)$ and consider $\boldsymbol{\alpha} \in \{(0,0),(1,1),(1,0),(0,1)\}$; then on Γ_{k-1} we have $\phi_k(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \mathbf{x}) = \phi_{k-1}(2\boldsymbol{\theta}^{(0,0)}, \mathbf{x})$. The four components $\phi_k(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot)$ are called harmonics. Their span is denoted with

$$E_k^{\theta} = span[\phi_k(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \cdot) : \boldsymbol{\alpha} \in \{(0,0),(1,1),(1,0),(0,1)\}].$$

Purpose of this analysis is to investigate the action of the smoothing and coarse-grid correction operators on couples (e_y, e_p) defined by

$$e_y(\mathbf{x}) = \sum_{\boldsymbol{\alpha},\boldsymbol{\theta}} Y_{\boldsymbol{\alpha},\boldsymbol{\theta}}\, \phi_k(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \mathbf{x}) \quad \text{and} \quad e_p(\mathbf{x}) = \sum_{\boldsymbol{\alpha},\boldsymbol{\theta}} P_{\boldsymbol{\alpha},\boldsymbol{\theta}}\, \phi_k(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \mathbf{x}).$$

Here (e_y, e_p) represent the error functions for y_h and p_h and $W_{\boldsymbol{\alpha},\boldsymbol{\theta}} = (Y_{\boldsymbol{\alpha},\boldsymbol{\theta}}, P_{\boldsymbol{\alpha},\boldsymbol{\theta}})$ denote the corresponding Fourier coefficients. With this decomposition of the error, the action of one smoothing step can be expressed as $W^{(1)}_{\boldsymbol{\alpha},\boldsymbol{\theta}} = \hat{S}(\boldsymbol{\alpha},\boldsymbol{\theta})\, W^{(0)}_{\boldsymbol{\alpha},\boldsymbol{\theta}}$ where $\hat{S}(\boldsymbol{\alpha},\boldsymbol{\theta})$ is the Fourier symbol [49] of the smoothing operator. To determine $\hat{S}(\boldsymbol{\alpha},\boldsymbol{\theta})$, recall that the functions $\phi_k(\boldsymbol{\theta}^{\boldsymbol{\alpha}}, \mathbf{x})$ are eigenfunctions of any discrete operator described by a difference stencil on the Γ_k grid. Therefore we have $S_k\phi_k(\boldsymbol{\theta}, \mathbf{x}) = \hat{S}_k(\boldsymbol{\theta})\, \phi_k(\boldsymbol{\theta}, \mathbf{x})$ that is, the symbol of S_k is its (formal) eigenvalue.

Now, consider one collective Gauss-Seidel iteration step applied to our model problem

$$\Delta_h y_h - p_h/\nu = f_h, \tag{78}$$

$$\Delta_h p_h + y_h = z_h. \tag{79}$$

In this case, one smoothing step at $\mathbf{x}$ corresponds to an update which sets the residuals at $\mathbf{x}$ equal to zero. In terms of the generic Fourier mode $\boldsymbol{\theta}$, $\hat{S}_k(\boldsymbol{\theta})$ is given by

$$\begin{bmatrix} (e^{-i\theta_1} + e^{-i\theta_2} - 4) & -h_k^2/\nu \\ h_k^2 & (e^{-i\theta_1} + e^{-i\theta_2} - 4) \end{bmatrix}^{-1} \times \begin{bmatrix} -(e^{i\theta_1} + e^{i\theta_2}) & 0 \\ 0 & -(e^{i\theta_1} + e^{i\theta_2}) \end{bmatrix}.$$

Clearly, $\hat{S}(\boldsymbol{\alpha}, \boldsymbol{\theta})$ consists of $\hat{S}_k(\boldsymbol{\theta}^{\boldsymbol{\alpha}})$ with $\boldsymbol{\alpha} \in \{(0,0),(1,1),(1,0),(0,1)\}$.

The smoothing property of S_k measures the action of this iteration on the high-frequency error components and can be defined as follows

$$\mu(S_k) = \sup\left\{r(\hat{S}_k(\boldsymbol{\theta})) : \boldsymbol{\theta} \in \{\boldsymbol{\theta}^{(1,1)}, \boldsymbol{\theta}^{(1,0)}, \boldsymbol{\theta}^{(0,1)}\}\right\}, \qquad (80)$$

where r denotes the spectral radius and $\mu(\cdot)$ is the smoothing factor which can be evaluated for any choice of values of h and ν.

The next step is to construct the Fourier symbol of the twolevel coarse-grid correction operator

$$CG_k^{k-1} = [I_k - I_{k-1}^k \, (A_{k-1})^{-1} \, I_k^{k-1} \, A_k].$$

We denote the corresponding symbol by

$$\widehat{CG}_k^{k-1}(\boldsymbol{\theta}) = [\hat{I}_k - \hat{I}_{k-1}^k(\boldsymbol{\theta}) \, (\widehat{A}_{k-1}(2\boldsymbol{\theta}))^{-1} \, \hat{I}_k^{k-1}(\boldsymbol{\theta}) \, \widehat{A}_k(\boldsymbol{\theta})].$$

The symbol of the coarse grid operator $\widehat{A}_{k-1}(\boldsymbol{\theta})$ is

$$\begin{bmatrix} \frac{2(\cos(2\theta_1)+\cos(2\theta_2))-4}{h_{k-1}^2} & -1/\nu \\ 1 & \frac{2(\cos(2\theta_1)+\cos(2\theta_2))-4}{h_{k-1}^2} \end{bmatrix},$$

and similarly one constructs $\widehat{A}_k(\boldsymbol{\theta})$ corresponding to the four harmonics, that is,

$$\begin{bmatrix} l(\boldsymbol{\theta}^{(0,0)}) & 0 & 0 & 0 & -1/\nu & 0 & 0 & 0 \\ 0 & l(\boldsymbol{\theta}^{(1,1)}) & 0 & 0 & 0 & -1/\nu & 0 & 0 \\ 0 & 0 & l(\boldsymbol{\theta}^{(1,0)}) & 0 & 0 & 0 & -1/\nu & 0 \\ 0 & 0 & 0 & l(\boldsymbol{\theta}^{(0,1)}) & 0 & 0 & 0 & -1/\nu \\ 1 & 0 & 0 & 0 & l(\boldsymbol{\theta}^{(0,0)}) & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & l(\boldsymbol{\theta}^{(1,1)}) & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & l(\boldsymbol{\theta}^{(1,0)}) & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & l(\boldsymbol{\theta}^{(0,1)}) \end{bmatrix},$$

where

$$l(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) = \frac{2(\cos(\theta_1^{\alpha_1}) + \cos(\theta_2^{\alpha_2})) - 4}{h_k^2}.$$

The symbol of the restriction operator is (here the hat denotes the Fourier symbol, not the injection operator)

$$\hat{I}_k^{k-1}(\boldsymbol{\theta}) = \begin{bmatrix} I(\boldsymbol{\theta}^{(0,0)}) & I(\boldsymbol{\theta}^{(1,1)}) & I(\boldsymbol{\theta}^{(1,0)}) & I(\boldsymbol{\theta}^{(0,1)}) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I(\boldsymbol{\theta}^{(0,0)}) & I(\boldsymbol{\theta}^{(1,1)}) & I(\boldsymbol{\theta}^{(1,0)}) & I(\boldsymbol{\theta}^{(0,1)}) \end{bmatrix},$$

where

$$I(\boldsymbol{\theta}^{\boldsymbol{\alpha}}) = \frac{1}{4}(1+\cos(\theta_1^{\alpha_1}))(1+\cos(\theta_2^{\alpha_2})).$$

For the prolongation operator we have $\hat{I}_{k-1}^k(\boldsymbol{\theta}) = \hat{I}_k^{k-1}(\boldsymbol{\theta})^T$.

Finally, the symbol of the twolevel method is given by

$$\widehat{TG}_k^{k-1}(\boldsymbol{\theta}) = \hat{S}_k(\boldsymbol{\theta})^{\nu_2}\, \widehat{CG}_k^{k-1}(\boldsymbol{\theta})\, \hat{S}_k(\boldsymbol{\theta})^{\nu_1}.$$

This is an 8×8 matrix corresponding to the four pairs $(Y_{\boldsymbol{\alpha},\boldsymbol{\theta}}, P_{\boldsymbol{\alpha},\boldsymbol{\theta}})$, $\boldsymbol{\alpha} \in \{(0,0),(1,1),(1,0),(0,1)\}$. In this framework the convergence factor is defined as follows

$$\eta(TG_k^{k-1}) = \sup\{r(\widehat{TG}_k^{k-1}(\boldsymbol{\theta})) : \boldsymbol{\theta} \in [-\pi/2,\pi/2)^2\}.$$

We can state the following theorem.

Theorem 6 *Under the assumption that all multilevel components are linear and that $(A_{k-1})^{-1}$ exists and $\hat{\mathcal{S}}_k(\boldsymbol{\theta}) : E_k^\theta \times E_k^\theta \to E_k^\theta \times E_k^\theta$ for all $\boldsymbol{\theta} \in [-\pi/2,\pi/2)^2$, we have a representation of the twolevel operator TG_k^{k-1} on $E_k^\theta \times E_k^\theta$ by a 8×8 matrix given by*

$$\widehat{TG}_k^{k-1}(\boldsymbol{\theta}) = \hat{\mathcal{S}}_k(\boldsymbol{\theta})^{\nu_2}\, \widehat{CG}_k^{k-1}(\boldsymbol{\theta})\, \hat{\mathcal{S}}_k(\boldsymbol{\theta})^{\nu_1},$$

and for given mesh with mesh-size h_k and given weight of the cost ν, the convergence factor estimate for the twolevel scheme applied to (78)-(79) is given by

$$\eta(TG_k^{k-1}) = \sup\{r(\widehat{TG}_k^{k-1}(\boldsymbol{\theta})) : \boldsymbol{\theta} \in [-\pi/2,\pi/2)^2\}.$$

We complete this section by reporting in Table 3 the values of $\eta(TG_k^{k-1})$ and those of $\mu(S_k)$ obtained with the twolevel analysis described above. For comparison, the observed value of convergence factor defined as the "asymptotic" value of the ratio between the discrete

Table 3: Convergence factors and smoothing factors.

	Local Fourier analysis		Experim.
(ν_1, ν_2)	$\mu(S_k)^{\nu_1+\nu_2}$	$\eta(TG_k^{k-1})$	$V(\nu_1, \nu_2)$
(1,1)	0.25	0.25	0.30
(2,1)	0.125	0.12	0.12
(2,2)	0.06	0.08	0.08
(3,2)	0.03	0.06	0.06
(3,3)	0.01	0.05	0.05

L^2 norms of residuals resulting from two successive multilevel cycles on the finest mesh is reported. Notice that the values reported in Table 3 are typical of the standard Poisson model problem. These values have been obtained considering the mesh size value h ranging in the interval $[0.01, 0.25]$ corresponding to the interval of mesh sizes used in the multilevel code. The value of the weight ν has been taken in the interval $[10^{-6}, 1]$.

4.7 Parabolic optimal control problems

We describe space-time multilevel schemes for the solution of parabolic optimal control problems in the whole space-time cylinder. The advantage of this approach, in contrast to the sequential one, is the ability to implement time coupling in the optimality system consisting of parabolic partial differential equations with opposite time orientation. For this purpose, appropriate collective smoothing schemes are defined. The space-time collective smoothing multigrid (CSMG) strategy results in fast solvers whose convergence factors are mesh independent and do not deteriorate as the weight of the cost of the control tends to be small.

Consider the following optimal control problem

$$\begin{cases} \min_{u\in L^2(Q)} J(y,u), & \\ \quad -\partial_t y + \sigma\,\Delta y &= u & \text{in } Q=\Omega\times(0,T), \\ \quad\quad y(\mathbf{x},0) &= y_0(\mathbf{x}) & \text{in } \Omega \text{ at } t=0, \\ \quad\quad y(\mathbf{x},t) &= 0 & \text{on } \Sigma=\partial\Omega\times(0,T), \end{cases} \tag{81}$$

where we take $y_0(\mathbf{x}) \in H_0^1(\Omega)$. Control may be required to track a desired trajectory given by $y_d(\mathbf{x},t) \in L^2(Q)$ or to reach a desired terminal state $y_T(\mathbf{x}) \in L^2(\Omega)$. For this purpose we choose a cost functional of the tracking type given by

$$J(y,u) = \frac{\alpha}{2}||y-y_d||^2_{L^2(Q)} + \frac{\beta}{2}||y(\cdot,T)-y_T||^2_{L^2(\Omega)} + \frac{\nu}{2}||u||^2_{L^2(Q)}. \tag{82}$$

Then there exists a unique solution to the optimal control problem above; see [36]. Here, $\nu > 0$ is the weight of the cost of the control and $\alpha \geq 0$, $\beta \geq 0$, $\alpha+\beta > 0$ are optimization parameters. For example, the case $\alpha = 1$, $\beta = 0$ corresponds to tracking without terminal observation.

The solution to (81) is characterized by the following optimality system

$$-\partial_t y + \sigma\,\Delta y = u, \tag{83}$$

$$\partial_t p + \sigma\,\Delta p + \alpha\,(y-z) = 0, \tag{84}$$

$$\nu u - p = 0, \tag{85}$$

with initial condition $y(\mathbf{x},0) = y_0(\mathbf{x})$ for the state equation (evolving forward in time) and terminal condition

$$p(\mathbf{x},T) = \beta(y(\mathbf{x},T) - y_T(\mathbf{x})), \tag{86}$$

for the adjoint equation (evolving backward in time).

Now, we discuss the design of two robust collective smoothing schemes for solving (83)–(86) discretized by finite differences and backward Eu-

ler scheme. For simplicity of illustration, we eliminate the control variable by means of the optimality condition $\nu u_h^m - p_h^m = 0$. We have

$$\begin{aligned} -\left[1+4\sigma\gamma\right] y_{i\,j\,m} \;&+\; \sigma\gamma\left[y_{i+1\,j\,m} + y_{i-1\,j\,m} + y_{i\,j+1\,m} + y_{i\,j-1\,m}\right] + y_{i\,j\,m-1} \\ &-\; \frac{\delta t}{\nu}\, p_{i\,j\,m} = 0, \quad 2 \le m \le N_t + 1, \end{aligned} \tag{87}$$

$$\begin{aligned} -\left[1+4\sigma\gamma\right] p_{i\,j\,m} \;&+\; \sigma\gamma\left[p_{i+1\,j\,m} + p_{i-1\,j\,m} + p_{i\,j+1\,m} + p_{i\,j-1\,m}\right] + p_{i\,j\,m+1} \\ &+\; \delta t\,\alpha\left(y_{i\,j\,m} - y_{d\,i\,j\,m}\right) = 0, \quad 1 \le m \le N_t. \end{aligned} \tag{88}$$

Here, $\gamma = \delta t/h^2$, and $t_m = (m-1)\delta t$, $m = 1, 2, \ldots, N_t + 1$.

Let us define a collective iteration step which is applied at any space-time grid point to update $w_{i\,j\,m} = (y_{i\,j\,m}, p_{i\,j\,m})$. For this purpose consider (87) and (88) for the two variables $y_{i\,j\,m}$ and $p_{i\,j\,m}$ at the grid point $i\,j\,m$. We can refer to the left-hand sides of (87) and (88) as the negative of the residuals $r_y(w_{i\,j\,m})$ and $r_p(w_{i\,j\,m})$, respectively. A step of a collective smoothing iteration at this point consists of a local update given by

$$\begin{pmatrix} y \\ p \end{pmatrix}^{(1)}_{i\,j\,m} = \begin{pmatrix} y \\ p \end{pmatrix}^{(0)}_{i\,j\,m} + \begin{bmatrix} -(1+4\sigma\gamma) & -\delta t/\nu \\ \delta t\,\alpha & -(1+4\sigma\gamma) \end{bmatrix}^{(0)\,-1}_{i\,j\,m} \begin{pmatrix} r_y \\ r_p \end{pmatrix}_{i\,j\,m} \tag{89}$$

where r_y and r_p denote the residuals at $i\,j\,m$ prior to the update. While a sweep of this smoothing iteration can be performed in any ordering of i, j in space, the problem of how to proceed along time direction arises.

To solve this problem we define a Gauss-Seidel-type update providing that the opposite time orientation of the state equation and of the adjoint equation is taken into account. For this purpose, to update the state variable we use the first vector component of (89) marching in the forward direction and the adjoint variable p is being updated using the second component of (89) marching backwards in time. In this way a robust iteration is obtained given by the following algorithm.

Algorithm 8 *Time-Splitted Collective Gauss-Seidel Iteration (TS-CGS)*

1. *Set the starting approximation.*
2. *For* $m = 2, \ldots, N_t$ *do*
3. *For* ij *in, e.g., lexicographic order do*

$$y^{(1)}_{i\,j\,m} = y^{(0)}_{i\,j\,m} + \frac{[-(1+4\sigma\gamma)]\, r_y(w) + \frac{\delta t}{\nu}\, r_p(w)}{[-(1+4\sigma\gamma)]^2 + \frac{\delta t^2}{\nu}\alpha}\Big|^{(0)}_{i\,j\,m},$$

$$p^{(1)}_{i\,j\,N_t-m+2} = p^{(0)}_{i\,j\,N_t-m+2} + \frac{[-(1+4\sigma\gamma)]\, r_p(w) - \delta t\, \alpha\, r_y(w)}{[-(1+4\sigma\gamma)]^2 + \frac{\delta t^2}{\nu}\alpha}\Big|^{(0)}_{i\,j\,N_t-m+2};$$

4. *end.*

Results of local Fourier analysis show that the TS-CGS scheme has good smoothing properties, independently of the value of ν. This is also confirmed by results of numerical experiments. In the regime of small σ, however, the TS-CGS iteration cannot provide robust smoothing as we expect from our experience of multigrid methods for anisotropic problems [49]. To overcome this difficulty, block-relaxation of the variables that are strongly connected should be performed. For small σ this means solving for all the pairs of state and adjoint variables along the time-direction for each space coordinate.

To describe the resulting procedure, consider the discrete optimality system (87)–(88) at any i, j and for all time steps. For each spatial grid point i, j a block-tridiagonal system is obtained, where each block is a 2×2 matrix corresponding to the pair (y, p) at a given time step. This block-tridiagonal system has the following form

$$M = \begin{bmatrix} A_2 & C_2 & & & & \\ B_3 & A_3 & C_3 & & & \\ & B_4 & A_4 & C_4 & & \\ & & & & & \\ & & & & C_{N_t} & \\ & & & B_{N_t+1} & A_{N_t+1} & \end{bmatrix}. \tag{90}$$

Centered at t_m, the entries B_m, A_m, C_m refer to the variables (y, p) at t_{m-1}, t_m, and t_{m+1}, respectively. The block A_m, $m = 2, \ldots, N_t$, is given by

$$A_m = \begin{bmatrix} -(1+4\sigma\gamma) & -\frac{\delta t}{\nu} \\ \\ \delta t\, \alpha & -(1+4\sigma\gamma) \end{bmatrix}, \tag{91}$$

where all functions within the brackets $[\,]$ are evaluated at t_m. Correspondingly, the B_m and C_m blocks are given by

$$B_m = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \text{ and } C_m = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}. \tag{92}$$

Clearly, for each time step, the variables neighboring the point ij are taken as constant and contribute to the right-hand side of the system.

It remains to discuss the block A_{N_t+1} for $\beta \neq 0$. At $t_m = T$, we have the terminal condition (86) which we rewrite as

$$\beta\,(y_h^m - y_{Th}^m) - p_h^m = 0, \quad m = N_t + 1.$$

Thus, the block A_{N_t+1} is given by

$$A_{N_t+1} = \begin{bmatrix} -(1+4\sigma\gamma) & -\frac{\delta t}{\nu} \\ \beta & -1 \end{bmatrix}. \tag{93}$$

For each i, j we have to solve a tridiagonal system $Mw = r$ where $w = (y_h^2, p_h^2, \ldots, y_h^{N_t+1}, p_h^{N_t+1})$ and $r = (r_y(w^2), r_p(w^2), \ldots, r_y(w^{N_t+1}), r_p(w^{N_t+1}))$. In particular we have $r_p(w^{N_t+1}) = p_h^{N_t+1} - \beta\,(y_h^{N_t+1} - y_{Th}^{N_t+1})$. Block-tridiagonal systems can be solved efficiently with $\mathcal{O}(N_t)$ effort. Summarizing the collective t-line relaxation is given by the following algorithm.

Algorithm 9 *Time-Line Collective Gauss-Seidel Iteration (TL-CGS)*

1. *Set the starting approximation.*

2. For ij in, e.g., lexicographic order do

$$\begin{pmatrix} y \\ p \end{pmatrix}^{(1)}_{ij} = \begin{pmatrix} y \\ p \end{pmatrix}^{(0)}_{ij} + M^{-1} \begin{pmatrix} r_y \\ r_p \end{pmatrix}_{ij};$$

3. end.

Also in this case r_y and r_p denote the residuals at i, j and for all m prior to the update. Since the solution in time is exact, no time splitting is required.

4.8 Local Fourier smoothing analysis

In this section, we perform local Fourier analysis of the TS-CGS and TL-CGS iterative schemes. The resulting estimates are in agreement with the observed computational behavior.

For simplicity, consider one space dimension. On the fine grid, consider the Fourier components $\phi(\boldsymbol{j}, \boldsymbol{\theta}) = e^{i\boldsymbol{j}\cdot\boldsymbol{\theta}}$ where i is the imaginary unit, $\boldsymbol{j} = (j_x, j_t) \in \boldsymbol{Z} \times \boldsymbol{Z}$, $\boldsymbol{\theta} = (\theta_x, \theta_t) \in [-\pi, \pi)^2$, and $\boldsymbol{j} \cdot \boldsymbol{\theta} = j_x\theta_x + j_t\theta_t$.

In a semicoarsening setting, the frequency domain is spanned by the following two sets of frequencies

$$\begin{aligned} \phi \text{ low frequency component} &\iff \theta_x \in [-\tfrac{\pi}{2}, \tfrac{\pi}{2}), \quad \theta_t \in [-\pi, \pi), \\ \phi \text{ high frequency component} &\iff \theta_x \in [-\pi, \pi) \setminus [-\tfrac{\pi}{2}, \tfrac{\pi}{2}), \quad \theta_t \in [-\pi, \pi). \end{aligned}$$

Let $\tilde{w}(\boldsymbol{j}) = (\tilde{y}(\boldsymbol{j}), \tilde{p}(\boldsymbol{j})) = \sum_{\boldsymbol{\theta}} \tilde{W}_{\boldsymbol{\theta}}\, \phi(\boldsymbol{j}, \boldsymbol{\theta})$ denotes the errors on the space-time grid and $\tilde{W}_{\boldsymbol{\theta}} = (\tilde{Y}_{\boldsymbol{\theta}}, \tilde{P}_{\boldsymbol{\theta}})$ are the corresponding Fourier coefficients. The action of one smoothing step on $\tilde{w}$ can be expressed by $\tilde{W}^{(1)}_{\boldsymbol{\theta}} = \hat{S}(\boldsymbol{\theta})\, \tilde{W}^{(0)}_{\boldsymbol{\theta}}$.

Now consider applying the TS-CGS step for solving a distributed

control problem with tracking. Substituting $\tilde{w}$ in (87)-(88) we obtain

$$\begin{pmatrix} -(1+2\sigma\gamma)+\sigma\gamma e^{-i\theta_x} & -\frac{\delta t}{\nu} \\ \alpha\delta t & -(1+2\sigma\gamma)+\sigma\gamma e^{-i\theta_x} \end{pmatrix} \begin{pmatrix} \tilde{Y}^{(1)}_{\boldsymbol{\theta}} \\ \tilde{P}^{(1)}_{\boldsymbol{\theta}} \end{pmatrix} = \begin{pmatrix} -(e^{-i\theta_t}+\sigma\gamma e^{i\theta_x}) & 0 \\ 0 & -(e^{i\theta_t}+\sigma\gamma e^{i\theta_x}) \end{pmatrix} \begin{pmatrix} \tilde{Y}^{(0)}_{\boldsymbol{\theta}} \\ \tilde{P}^{(0)}_{\boldsymbol{\theta}} \end{pmatrix}.$$

Hence

$$\hat{S}(\boldsymbol{\theta}) = \begin{pmatrix} -(1+2\sigma\gamma)+\sigma\gamma e^{-i\theta_x} & -\frac{\delta t}{\nu} \\ \alpha\delta t & -(1+2\sigma\gamma)+\sigma\gamma e^{-i\theta_x} \end{pmatrix}^{-1} \times \begin{pmatrix} -(e^{-i\theta_t}+\sigma\gamma e^{i\theta_x}) & 0 \\ 0 & -(e^{i\theta_t}+\sigma\gamma e^{i\theta_x}) \end{pmatrix}. \tag{94}$$

In Figure 8 (left) we depict the smoothing factor of the TS-CGS scheme as a function of ν and γ. It appears that μ is independent of the value of the weight ν and of the discretization parameter γ, as long as γ is sufficiently large. For $\gamma \to 0$ or $\sigma \to 0$ and moderate values of ν, worsening of the smoothing factor can be observed. Similar results are obtained with different choices of the time-step size and in case $\alpha = 0$, $\beta \neq 0$.

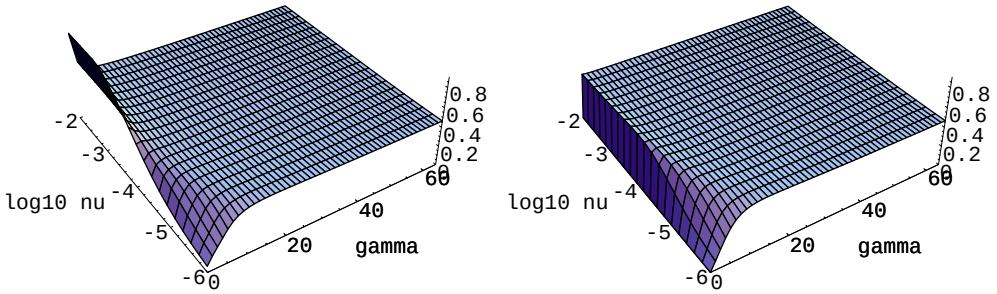

Figure 8: Smoothing factors of TS-CGS (left) and TL-CGS (right) schemes as functions of ν and γ; $\delta t = 1/64$, $\alpha = 1$, and $\sigma = 1$.

Next consider the case of TL-CGS relaxation. The Fourier symbol of the smoothing operator is given by the following 2×2 matrix

$$\hat{S}(\boldsymbol{\theta}) = -(A + B\, e^{-i\theta_t} + C\, e^{i\theta_t} + \tilde{I}\, e^{-i\theta_x})^{-1}(\tilde{I}\, e^{i\theta_x}),$$

where

$$A = \begin{bmatrix} -(1+2\sigma\gamma) & -\frac{\delta t}{\nu} \\ \delta t\alpha & -(1+2\sigma\gamma) \end{bmatrix},\ B_m = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \text{ and } C_m = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix},$$

and $\tilde{I} = \sigma\gamma\, I$, I is the 2×2 identity matrix.

In Figure 8 (right), the smoothing factor of the TL-CGS scheme as a function of ν and γ is shown. Notice that the smoothing factor of this scheme is independent of ν and γ. For $\sigma = 0$ no spatial coupling is present and the TL-CGS scheme becomes an exact solver, i.e. $\mu = 0$ results.

4.9 Receding horizon approach

Results of numerical experiments and Fourier analysis estimates demonstrate the ability of the multigrid schemes presented here in solving tracking and terminal observation optimal control problems. This fact suggests combining these multigrid schemes with receding horizon techniques [34] to develop an efficient (sub) optimal control algorithm for tracking a desired trajectory over very long time intervals. In the following, we sketch the implementation of the multigrid receding horizon scheme.

Consider the optimal control problem of tracking y_d for $t \geq 0$. Define time windows of size Δt. In each time window, an optimal control problem with tracking ($\alpha = 1$) and terminal observation ($\beta = 1$) is solved. The resulting optimal state at $n\Delta t$ defines the initial condition for the next optimal control problem defined in $(n\Delta t, (n+1)\Delta t)$ with desired terminal state given by $y_T(\mathbf{x}) = y_d(\mathbf{x}, (n+1)\Delta t)$. The following algorithm results.

Algorithm 10 *Multigrid Receding Horizon Scheme (MG-RH)*

1. *Set* $y(\mathbf{x}, 0) = y_0(\mathbf{x})$ *and* $n = 0$.
2. *Set* $y_T(\mathbf{x}) = y_d(\mathbf{x}, (n+1)\Delta t)$.
3. *CSMG Solve (83)–(85) in* $(n\Delta t, (n+1)\Delta t)$.
4. *Update* $n := n + 1$, *set* $y_0(\mathbf{x}) = y(\mathbf{x}, n\Delta t)$ *and goto 2.*

5 Globalization issues

In a convex setting where the optimal control solution is unique, solving the optimality system is equivalent to solving the optimal control problem. However, in general, these solutions represent only extremal points and additional conditions must be satisfied to guarantee that they are the minima sought. We now consider optimal control problems that possibly have multiple extremal points and describe a multigrid method [9] of how to escape undesired maxima or saddle points. For ease of reading, we recall the content of Section 4.1.

Consider the optimal control problem

$$\begin{cases} \min_{u\in U} J(y,u), \\ \qquad e(y,u) \;=\; 0 \quad \text{in } \Omega, \end{cases} \tag{95}$$

where y and u denote the state- and control variables of a controlled partial differential equation expressed as $e(y,u)=0$, with $e : Y \times U \to Z$ for appropriate Hilbert spaces Y, U, and Z. Ω is an open bounded set in $\mathbf{R}^d$. The cost functional J is formally given by

$$J(y,u) = h(y) + \nu\, g(u), \tag{96}$$

where $\nu > 0$ is the weight of the cost of the control. Here g and h are required to be continuously differentiable, bounded from below, and such that $g(u) \to \infty$ as $||u|| \to \infty$. Allowing g and h to be locally non-convex and e to be possibly nonlinear, (95) may have multiple extremals including minima, maxima, and saddle points.

Local minima satisfy the first-order necessary conditions. To define these conditions consider the Lagrangian

$$L(y,u,p) = J(y,u) + \langle e(y,u), p\rangle_{Z,Z^*},$$

where p is the Lagrange multiplier, the adjoint variable. By equating to zero the Frechét derivatives of L with respect to the triple (y,u,p),

we obtain the following optimality system

$$\begin{aligned} e(y,u) &= 0, \\ e_y(y,u)^*\, p &= -h'(y), \\ \nu\, g'(u) + e_u^*\, p &= 0. \end{aligned} \tag{97}$$

Numerical approximations to solutions of (97) can be obtained, after discretization, using multilevel or other iterative methods starting from any initial guess. The particular choice of the starting approximation determines towards which solution the iterative scheme will converge. Solutions to (97) are not necessarily local minima, rather they are extremal points.

Our purpose is to introduce in the multilevel scheme a mechanism allowing to distinguish among different types of extremal points and providing the direction for escaping undesired maxima and saddle points.

5.1 Second-order conditions for a minimum

If J and e are twice continuously differentiable, the second-order sufficient conditions for a minimum are given by (97) and the following

$$L_{xx}(y,u,p)(v,v) \geq c_1\, ||v||^2, \quad c_1 > 0, \text{ for all } v \in \mathcal{N}(e'(y,u)), \tag{98}$$

where $x = (y,u)$ and e' represents the linearized equality constraint; see, e.g., [20]. We assume that the null space $\mathcal{N}(e'(y,u))$ can be represented by $\mathcal{N}(e'(y,u)) = T(y,u)\, U$, where

$$T(y,u) = \begin{bmatrix} -e_y^{-1}\, e_u \\ I_u \end{bmatrix},$$

and e_y, e_u are evaluated at (y,u). Therefore condition (98) becomes

$$H(y,u,p)(w,w) \geq c_2\, ||w||^2, \quad c_2 > 0, \tag{99}$$

for all $w \in U$. The operator H is the reduced Hessian defined by

$$H(y,u,p) = T(y,u)^*\, L_{xx}(y,u,p)\, T(y,u).$$

That is, H is given by

$$H(y,u,p) = L_{uu}(y,u,p) + C(y,u)^* \, L_{yy}(y,u,p) \, C(y,u), \qquad (100)$$

where $C(y,u) = e_y(y,u)^{-1} \, e_u(y,u)$, assuming $e_{yu}(y,u) = 0$.

Notice that H is symmetric. Therefore condition (99) requires that, in order to have a minimum, all eigenvalues of the reduced Hessian be positive. Otherwise, the occurrence of nonpositive eigenvalues indicates the presence of possible maxima or saddle points. Thus, in principle, once a solution to (97) is found, one should solve the eigenvalue problem associated to H. If all eigenvalues are positive, we have a minimum and therefore a solution to (95). If some eigenvalues are negative, the solution of the optimality system is not a solution to the optimal control problem.

Clearly, in an infinite dimensional setting, the analysis of the spectrum of H may be an overwhelming task. Even after discretization, solving the eigenvalue problem may be computationally more expensive than solving the optimality system.

The multilevel strategy provides a way to overcome this difficulty. A successful multilevel procedure is based on a hierarchy of discrete equations able to represent, at different scales, the underlying continuous problem. We make the assumption that the spectral properties of the reduced Hessian are well represented on the hierarchy of grids and therefore we can define a globalization step based on the spectral properties of the Hessian H on the coarsest grid. In the case negative eigenvalues of the reduced Hessian are detected, we use the eigenvector corresponding to the smallest eigenvalue to determine an escape direction. This direction of negative curvature [39, 42] is given by the eigenvector corresponding to the negative eigenvalue with largest absolute value.

If such an eigenvalue exists, the normalized eigenvector ϕ_h is used

to perform the following globalization step

$$u_h^{new} = u_h - \sigma\,\phi_h. \tag{101}$$

We choose $|\sigma| = \sqrt{\beta}$ (with β as in (103); see [42]) and the sign of σ is such that $\sigma\,\phi_h \cdot (\nu g'(u_h) + e_u^*\, p_h(u_h)) \geq 0$.

Once we escape the undesired critical point (at the coarsest grid), the multigrid procedure continues as described choosing components that are minimizing.

In the above discussion, we tacitly assumed that the spectrum of the Hessian before discretization also consists of pure point-spectrum. This is the case, for example, if e_y^{-1} is a compact operator corresponding to the case of elliptic- or parabolic-type partial differential equations and $g(u) = \frac{1}{2}||u||^2$. A similar remark applies, for example, for the choice $g(u) = \frac{1}{2}||\nabla u||^2$. In either of these two cases, the resulting reduced Hessian is such that its spectrum can be well represented in a hierarchy of grids.

5.2 Globalization of the FAS scheme

The present globalization approach relies on two features. First, the FAS multilevel procedure is defined such that it provides a descent step for the optimal control constrained minimization problem. Second, on the coarsest grid one analyzes the possible encounter of extremal points that are not minima and define an escape direction, if necessary, on the basis of negative-curvature eigenvectors.

To guarantee a multilevel step which is minimizing, we define the smoothing process based on the gradient of the reduced cost functional and show that the FAS coarse-grid correction step provides a descent update.

To define the smoothing iteration S, the discretized state equation to obtain $y_h(\mathbf{x})$ as function of $u_h(\mathbf{x})$ at the grid point is used. Replacing y_h in the adjoint equation by this function, we obtain $p(\mathbf{x})$ as function

of $u(\mathbf{x})$. From these considerations, the optimality condition becomes

$$\nu\, g'(u_h) + e_u^*\, p_h(u_h) = 0. \tag{102}$$

This equation corresponds to requiring that the gradient of the reduced cost functional $J(y(u), u)$ with respect to the control variable u is zero. In general, in order to update the control function in the smoothing process, we use (102) to perform a few steps of the following descent scheme

$$u_h^{new} = u_h - \beta\,(\nu\, g'(u_h) + e_u^*\, p_h(u_h)). \tag{103}$$

An optimal choice of the scaling factor $\beta > 0$ may be done using line search methods.

Now consider the case $e(y, u) = -\Delta y - u$, we show that the FAS coarse-grid correction provides a descent direction in the sense that

$$(\nu\, g'(u_h) - p_h, I_H^h(u_H - \hat{I}_h^H u_h))_h < 0,$$

unless $u_H = \hat{I}_h^H u_h$, occurring at convergence.

Starting from an initial approximation and after a few pre-smoothing steps the resulting triple (y_h, u_h, p_h) satisfies the optimality system up to residuals (d_h^1, d_h^2, d_h^3), that is,

$$\begin{aligned} -\Delta_h y_h - u_h &= d_h^1, \\ -\Delta_h\, p_h + y_h - z_h &= d_h^2, \\ \nu\, g'(u_h) - p_h &= d_h^3. \end{aligned} \tag{104}$$

For the coarse-grid process, we take $\hat{I}_h^h = I_h^H$ where I_h^H is the full-weighting restriction operator. For I_H^h we choose bilinear interpolation which is the adjoint of the restriction operator just defined [27], i.e. $(I_h^H u_h, v_H)_H = (u_h, I_H^h v_H)_h$. Define $z_H = I_h^H z_h$. With this setting, we obtain the following coarse-grid FAS equations

$$\begin{aligned} -\Delta_H y_H - u_H &= I_h^H \Delta_h y_h - \Delta_H I_h^H y_h, \\ -\Delta_H p_H + y_H - z_H &= I_h^H \Delta_h p_h - \Delta_H I_h^H p_h, \\ \nu\, g'(u_H) - p_H &= 0. \end{aligned} \tag{105}$$

As usual in two-grid convergence analysis, we assume that this coarse system of equations is solved exactly. From the first equation of (105), and using the corresponding equation in (104) we obtain

$$u_H - I_h^H u_h = -\Delta_H(y_H - I_h^H y_h) + I_h^H d_h^1. \tag{106}$$

Combining the fine and coarse adjoint equations we have

$$p_H - I_h^H p_h = \Delta_H^{-1}(y_H - I_h^H y_h) + \Delta_H^{-1} I_h^H d_h^2. \tag{107}$$

Let us assume that

$$(g'(v_H) - I_h^H\, g'(v_h), v_H - I_h^H v_h)_H \geq \delta' \, \|v_H - I_h^H v_h\|_H^2, \tag{108}$$

for some $\delta' > 0$ independent of v_h and v_H. Note that (108) is satisfied, for example, if g' is linear or if I_h^H is strict injection and g is strictly convex.

With these preparations we are ready to show that the update step of the FAS coarse-grid correction follows a descent direction

$$\begin{aligned}
&(\nu\, g'(u_h) - \, p_h, I_H^h(u_H - I_h^H u_h))_h = (I_h^H(\nu\, g'(u_h) - \, p_h), u_H - I_h^H u_h)_H \\
&= (\nu I_h^H\, g'(u_h) - I_h^H\, p_h, u_H - I_h^H u_h)_H \\
&= (\nu I_h^H\, g'(u_h) - p_H + \Delta_H^{-1}(y_H - I_h^H y_h) + \Delta_H^{-1} I_h^H d_h^2, u_H - I_h^H u_h)_H \\
&= -\,\nu\,(g'(u_H) - I_h^H\, g'(u_h), u_H - I_h^H u_h)_H \\
&\quad +(\Delta_H^{-1}(y_H - I_h^H y_h) + \Delta_H^{-1} I_h^H d_h^2, -\Delta_H(y_H - I_h^H y_h) + I_h^H d_h^1)_H \\
&= -\,\nu\,(g'(u_H) - I_h^H\, g'(u_h), u_H - I_h^H u_h)_H - (y_H - I_h^H y_h, y_H - I_h^H y_h)_H \\
&\quad +(\Delta_H^{-1}(y_H - I_h^H y_h), I_h^H d_h^1)_H - (\Delta_H^{-1} I_h^H d_h^2, \Delta_H(y_H - I_h^H y_h))_H \\
&\quad +(\Delta_H^{-1} I_h^H d_h^2, I_h^H d_h^1)_H \\
&\leq -\nu\,(g'(u_H) - I_h^H\, g'(u_h), u_H - I_h^H u_h)_H \\
&\quad +\tfrac{1}{2}(\|\Delta_H^{-1} I_h^H d_h^1\|_H^2 + \|\Delta_H^{-1} I_h^H d_h^2\|_H^2 + \|I_h^H d_h^1\|_H^2 + \|I_h^H d_h^2\|_H^2) \\
&\leq -\nu\,\delta' \|u_H - I_h^H u_h\|_H^2 \\
&\quad +\tfrac{1}{2}(\|\Delta_H^{-1} I_h^H d_h^1\|_H^2 + \|\Delta_H^{-1} I_h^H d_h^2\|_H^2 + \|I_h^H d_h^1\|_H^2 + \|I_h^H d_h^2\|_H^2).
\end{aligned}$$

Therefore

$$(\nu\, g'(u_h) - \, p_h, I_H^h(u_H - I_h^H u_h))_h < 0,$$

if (108) holds and the residuals d_h^1 and d_h^2 are sufficiently small.

Finally we show that the coarse-grid correction step does not produce overshooting in the sense that $(\hat{J}'(u_h)_h, \hat{J}'(u_h^{new})_h)_h \geq 0$. We consider the case where $g'(u) = u$. We have the following

$$\begin{aligned}
& (\hat{J}'(u_h)_h, \hat{J}'(u_h^{new})_h)_h \\
= \; & (\nu\, u_h - p_h, \nu\,(u_h + I_H^h(u_H - I_h^H u_h)) - (p_h + I_H^h(p_H - I_h^H p_h))_h \\
= \; & \|\nu\, u_h - p_h\|_h^2 + (\nu\, u_h - p_h, I_H^h[\nu\,(u_H - I_h^H u_h) - (p_H - I_h^H p_h)])_h \\
= \; & \|\nu\, u_h - p_h\|_h^2 - \|I_h^H(\nu\, u_h - p_h)\|_H^2 \geq 0,
\end{aligned}$$

where we use $\|I_h^H\| \leq 1$.

6 Appendix: A 1D MG code for the Poisson problem in MATLAB

The following is a matlab multilevel algorithm for the solution of the one-dimensional model problem Poisson equation on $(0, 1)$ subject to Dirichlet boundary conditions; see [13, 14] for details.

```
%------------------- main ----------------------------------
% mgvee.m

clear

% % V-cycle scheme to solve
% %     - Delta u = f on [0,1]
% % with boundary values given by the function g(x).

% N is the number of subintervals

global N; N = 128; h = 1/N; tol = 10e-6;

% lmax determines the coarsest grid level; original grid = level 1
% e.g., lmax = 4 means one has to restrict
% to coarser grids 3 times
global lmax; lmax = 6;


% initial guess
j = 0:N; initguess = sin(20*pi*j*h); v = initguess;

%zeros(size(initguess));

% exact solution
vexact = sin(2*pi*j*h);

% right-hand side
f = 4*pi*pi*sin(2*pi*j*h);


% main engine
```

```
relaterr = 10; ctr = 0; rfin_norm_old=1.0; rfin_norm=1.0;

relaterr_old=1.0;


while rfin_norm > tol
    [vnew,rfin_norm] = vcycle (v, f, 1);


% some output info
    relaterr = norm(vexact-vnew,2)/(norm(vexact,2)+.1);
    fprintf('relative error is %6.10d\n',relaterr);

    conv_fact_r=rfin_norm/rfin_norm_old;
    rfin_norm_old=rfin_norm;
    fprintf('convergence factor based on residual is %6.10d\n',conv_fact_r);

    v = vnew;
    ctr = ctr +1;
end

fprintf('The norm of the solution error is %6.10d\n', norm(v-vexact,2))
fprintf('The norm of the solution residual is %6.10d\n', rfin_norm)
fprintf('The number of iterations required to satisfy tolerance is %d\n', ctr)


%-----------------------------------------------------
% vcycle.m


function [vcycleout,rfin_norm] = vcycle (v, f, L)
    global N;
    global lmax;

    % number of iterations in one level
    numiter = 2;
    v = wjacobi(v, f, numiter, L);

    % if not in yet the coarsest grid, restrict
    % if already in coarsest grid, relax and leave
    if L ~= lmax
```

```
  % should be rh instead of r2h, it is still fine grid
        r2h = compresidual (v,f,L);
        f2h = restrictfw (r2h,L);   %output is now at level L+1
  % zero initial guess
        v2h = zeros(size(f2h));
  % recursion:
        v2h_new = vcycle (v2h, f2h, L+1);
    else
        v = wjacobi (v, f, numiter, L);
        vcycleout = v; return
    end
  % prolongate error v2h_new
        errh = prolongate(v2h_new-v2h, L+1);    %output is now at level L
        v = v + errh;  %% error correction

    % relax and then leave
        v = wjacobi (v, f, numiter, L);
        vcycleout = v;

        rfin_norm = norm(compresidual(v,f,L),2);

% end

%-------------------------------------------------------
% wjacobi.m

function wjreturn = wjacobi(v, f, k, L)
    global N;

    n = N / 2^(L-1);    % size of the matrices
    w = 2/3;            % weight
    h = 2^(L-1) / N;    % size of the interval

    for i = 1:k
        tempans = .5 * ( v(1:n-1) + v(3:n+1) + h*h*f(2:n) );
        vtemp = v;
        vtemp(2:n) = tempans;                   % keep boundary values unchanged
        v = (1-w)*v + w*vtemp;                      % new iterate (weighted jacobi)
    end
    wjreturn = v;
% end
```

```
%-------------------------------------------------------
% compresidual.m

function residualout = compresidual (v, f, L)
    global N;
        n = N / 2^(L-1);    % size of the matrices
        h = 2^(L-1) / N;    % size of the interval
    vtemp = v;
    tempans = (v(1:n-1) + v(3:n+1) - 2*v(2:n) )/h/h;
    vtemp(2:n) = tempans;

    residualout = f + vtemp;
% end

%-------------------------------------------------------
% restrictfw.m

function restrictionout = restrictfw (r, L)
    % restriction by full weighting
    global N;
        n = N / 2^(L-1);    % size of the matrices
        c = r(1:2:end);  % c has size n/2 +1
    tempans = ( r(2:2:n-2) + r(4:2:n) + 2 * r(3:2:n-1) )/4;
    c(2:n/2) = tempans;
    restrictionout = c;
% end


%-------------------------------------------------------
% prolongate.m

function prolonged = prolongate(v, L)
    global N;
        n = N / 2^(L-1);    % size of the matrices
    c=zeros(1,2*n +1);
        c(1:2:end) = v;     % entries of v are transferred as they are
        c(2:2:end) = ( v(1:n) + v(2:n+1) )/2;
    prolonged = c;
% end
```

7 Appendix: A 2D MG code for the Poisson problem in FORTRAN

The following is a FORTRAN 77 multilevel algorithm for the solution of the Poisson equation on a rectangle subject to Dirichlet boundary conditions; see [13, 14] for details.

```
      PROGRAM CYCLEV
C
C    MULTI-GRID ALGORITHM FOR THE SOLUTION OF THE POISSON PROBLEM:
C              DELTA(U)=F
C
C     EXPLANATIONS OF PARAMETERS:
C----------------------------
C
C  NX1-    NUMBER OF INTERVALS INX-DIRECTION ON THE COARSEST GRID
C  NY1-    NUMBER OF INTERVALS IN Y-DIRECTION ON THE COARSEST GRID
C  H1-     LENGTH OF EACH INTERVAL
C  M-      NUMBER OF LEVELS
C  NU1- NUMBER OF RELAXATION SWEEPS IN EACH CYCLE BEFORE TRAN-
C  SFER TO THE COARSER GRID
C  NU2-    NUMBER OF SWEEPS IN EACH CYCLE AFTER COMING BACK FROM
C  THE COARSER GRID
C  NCYC-   NUMBER OF CYCLES
C  IFAS-   IFAS=1 FAS SCHEME, IFAS=0 MG SCHEME
C
C  G(X,Y)- BOUNDARY VALUES AND INITIAL APPROXIMATION
C  FOR THE SOLUTION U(X,Y) ARE GIVEN BY THE C FUNCTION G(X,Y)
C
C CORRECTION SCHEME BEGINS FROM THE FINEST GRID TO COARSEST GRID.
C
C
      implicit real*8 (a-h,o-z)
      EXTERNAL G,F,Z
      COMMON Q(18000)

      DIMENSION IST(200)
```

```
      DATA NX1/2/,NY1/2/,H1/0.5/,M/5/,NU1/2/,NU2/2/,NCYC/10/
c
c-----set method IFAS=1 nonlinear method, IFAS=0 linear method c
      IFAS=1
c
c-----set up: the grid
c
      DO 1 K=1,M
      K2=2**(K-1)
      CALL GRDFN(K,NX1*K2+1,NY1*K2+1,H1/K2)
      CALL GRDFN(K+M,NX1*K2+1,NY1*K2+1,H1/K2)
  1   CONTINUE
      WU=0.
c
c-----set up: the data (initial approx, rhs, bc, etc.) c
      CALL PUTF(M,G,0)
      CALL PUTB(M,G)
      CALL PUTF(2*M,F,2)

      ERRMX=1.0
      IREL=0
c
c-----start cycling
c
      DO 5 IC=1,NCYC
c-----store the previous relative error
      ERROLD=ERRMX
c-----go up
      DO 3 KM=1,M
      K=1+M-KM
c-----pre-smoothing, NU1 times
      DO 2 IR=1,NU1
    2 CALL RELAX(K,K+M,WU,M,ERRM)
c-----store the relative error
      IF(K.EQ.M) ERRMX=ERRM
c-----set initial zero approx. on the coarse grid
      IF (K.NE.M.AND.IFAS.EQ.0) CALL PUTZ(K)
c-----compute residual res=b-Au (and transfer it to k-1)

c      H   H   h   h  h
c     r = I  (f - L  u )
```

```
c          h

      IF(K.GT.1) CALL RESCAL(K,K+M,K+M-1)
c
c-----set initial approx. on the coarse grid
      IF (K.NE.1.AND.IFAS.EQ.1) CALL PUTU(K,K-1)
c-----compute the right-hand side

c      H   H   h   h  h     H  H h
c     f = I  (f - L  u ) + L (I u)
c          h                  h

      IF(K.GT.1.AND.IFAS.EQ.1) CALL CRSRES(K-1,K+M-1)
    3 CONTINUE
c
c-----go down
      DO 5 K=1,M
      DO 4 IR=1,NU2
    4 CALL RELAX(K,K+M,WU,M,ERRM)
c-----interpolate the coarse solution (error function)

c     to the next finer grid and add to the existing approximation

c

c      h   h    h  H   H h
c     u = u  + I (u - I u )
c

      IF(IFAS.EQ.1.AND.K.LT.M) CALL SUBTRT(K+1,K)
c
      IF(K.LT.M) CALL INTADD(K,K+1)
c
c-----compute the convergence factor using the relative c error

      IF(K.EQ.M) write(*,*) 'rho ', errmx/errold
   5  CONTINUE
C
C  PRINT THE SOLUTION (FINEST GRID) C
 999  CONTINUE
      OPEN(UNIT=17,FILE='TEST.DAT',STATUS='UNKNOWN')
```

```
      CALL KEY(M,IST,II,JJ,H)
      JSTEP=17
      IF(JJ.GT.9) JSTEP=INT(JJ/9.)+1
      DO 90 I=1,II
   90 WRITE(17,100) (Q(IST(I)+J),J=1,JJ,JSTEP)
c
c     calculate the L00 error c
      difmx=0.0
      do 95 i=1,ii
         x=(i-1)*h
         do 95 j=1,jj
            y=(j-1)*h
            err=abs(q(ist(i)+j)-g(x,y))
            difmx=max(difmx,err)
   95 continue
      write(*,*) 'l00 norm of the error =',difmx
  100 FORMAT(1X,257(1X,E8.2))
      STOP
      END
C
      REAL*8 FUNCTION F(X,Y)
      implicit real*8 (a-h,o-z)
      PI=4.0D0 * DATAN(1.0D0)
      PI2=PI*PI
      F=-(2.0*PI2)*SIN(PI*X)*SIN(PI*Y)
      RETURN
      END
C
      REAL*8 FUNCTION G(X,Y)
      implicit real*8 (a-h,o-z)
      PI=4.0D0 * DATAN(1.0D0)
      PI2=PI*PI
      G=SIN(PI*X)*SIN(PI*Y)
      RETURN
      END
C
      SUBROUTINE GRDFN(K,M,N,HH)
      implicit real*8 (a-h,o-z)
      COMMON/GRD/NST(20),IMX(20),JMX(20),H(20)
      DATA IQ/1/
      NST(K)=IQ
```

```
      IMX(K)=M
      JMX(K)=N
      H(K)=HH
      IQ=IQ+M*N
      RETURN
      END
C
      SUBROUTINE KEY(K,IST,M,N,HH)
      implicit real*8 (a-h,o-z)
      COMMON/GRD/NST(20),IMX(20),JMX(20),H(20)
      DIMENSION IST(200)
      M=IMX(K)
      N=JMX(K)
      IS=NST(K)-N-1
      DO 1 I=1,M
      IS=IS + N
    1 IST(I)=IS
      HH=H(K)
      RETURN
      END
C
      SUBROUTINE PUTF(K,F,NH)
      implicit real*8 (a-h,o-z)
      COMMON Q(18000)
      DIMENSION IST(200)
      CALL KEY (K,IST,II,JJ,H)
      H2=H**NH
      DO 1 I=1,II
       DO 1 J=1,JJ
         X=(I-1)*H
         Y=(J-1)*H
    1 Q(IST(I)+J)=F(X,Y)*H2
      RETURN
      END
C
      SUBROUTINE PUTZ(K)
      implicit real*8 (a-h,o-z)
      COMMON Q(18000)
      DIMENSION IST(200)
      CALL KEY(K,IST,II,JJ,H)
      DO 1 I=1,II
```

```
      DO 1 J=1,JJ
    1 Q(IST(I)+J)=0.
      RETURN
      END
C
      SUBROUTINE PUTU(KF,KC)
      implicit real*8 (a-h,o-z)
      COMMON Q(18000)
      DIMENSION IUF(200), IUC(200)
      CALL KEY(KF,IUF,IIF,JJF,HF)
      CALL KEY(KC,IUC,IIC,JJC,HC)
      DO 1 IC=1,IIC
      IF=2*IC-1
      IFO=IUF(IF)
      ICO=IUC(IC)
      JF=-1
      DO 1 JC=1,JJC
      JF=JF+2
      Q(ICO+JC)=Q(IFO+JF)
    1 CONTINUE
      RETURN
      END
C
      SUBROUTINE PUTB(K,F)
      implicit real*8 (a-h,o-z)
      COMMON Q(18000)
      DIMENSION IST(200)
      CALL KEY (K,IST,II,JJ,H)
      DO 1 I=1,II
      X=(I-1)*H
      Y=0.0
      Q(IST(I)+1)=F(X,Y)
      Y=(JJ-1)*H
      Q(IST(I)+JJ)=F(X,Y)
    1 CONTINUE
      DO 2 J=1,JJ
      Y=(J-1)*H
      X=0.0
      Q(IST(1)+J)=F(X,Y)
      X=(II-1)*H
      Q(IST(II)+J)=F(X,Y)
```

```
    2 CONTINUE
      RETURN
      END
C
      SUBROUTINE SUBTRT(KF,KC)
      implicit real*8 (a-h,o-z)
      COMMON Q(18000)
      DIMENSION IUF(200),IUC(200)
      CALL KEY(KF,IUF,IIF,JJF,HF)
      CALL KEY(KC,IUC,IIC,JJC,HC)
      DO 1 IC=1,IIC
      IF=2*IC-1
      IFO=IUF(IF)
      ICO=IUC(IC)
      JF=-1
      DO 1 JC=1,JJC
      JF=JF+2
      Q(ICO+JC)=Q(ICO+JC)-Q(IFO+JF)
    1 CONTINUE
      RETURN
      END
C
      SUBROUTINE INTADD(KC,KF)
      implicit real*8 (a-h,o-z)
      COMMON Q(18000)
      DIMENSION ISTC(200),ISTF(200)
      CALL KEY(KC,ISTC,IIC,JJC,HC)
      CALL KEY(KF,ISTF,IIF,JJF,HF)
      HF2=HF*HF
      DO 1 IC=2,IIC
      IF=2*IC-1
      JF=1
      IFO=ISTF(IF)
      IFM=ISTF(IF-1)
      ICO=ISTC(IC)
      ICM=ISTC(IC-1)
      DO 1 JC=2,JJC
      JF=JF+2
      A=.5*(Q(ICO+JC)+Q(ICO+JC-1))
      AM=.5*(Q(ICM+JC)+Q(ICM+JC-1))
      Q(IFO+JF) = Q(IFO+JF)+Q(ICO+JC)
```

```
      Q(IFM+JF) = Q(IFM+JF)+.5*(Q(ICO+JC)+Q(ICM+JC))
      Q(IFO+JF-1)=Q(IFO+JF-1)+A
    1 Q(IFM+JF-1)=Q(IFM+JF-1)+.5*(A+AM)
      RETURN
      END
C
      SUBROUTINE RESCAL(KF,KRF,KRC)
      implicit real*8 (a-h,o-z)
      COMMON Q(18000)
      DIMENSION IUF(200),IRF(200),IRC(200)
      CALL KEY(KF,IUF,IIF,JJF,HF)
      CALL KEY(KRF,IRF,IIF,JJF,HF)
      CALL KEY(KRC,IRC,IIC,JJC,HC)
      IIC1=IIC-1
      JJC1=JJC-1
      HF2=HF*HF
      DO 1 IC=2,IIC1
      ICR=IRC(IC)
      IF=2*IC-1
      JF=1
      IFR=IRF(IF)
      IFO=IUF(IF)
      IFM=IUF(IF-1)
      IFP=IUF(IF+1)
      DO 1 JC=2,JJC1
      JF=JF+2
      S=Q(IFO+JF+1)+Q(IFO+JF-1)+Q(IFM+JF)+Q(IFP+JF)
    1 Q(ICR+JC)=4.*(Q(IFR+JF)-S+4.*Q(IFO+JF))
      RETURN
      END
C
      SUBROUTINE CRSRES(K,KRHS)
      implicit real*8 (a-h,o-z)
      COMMON Q(18000)
      DIMENSION IST(200),IRHS(200)
      CALL KEY(K,IST,II,JJ,H)
      CALL KEY(KRHS,IRHS,II,JJ,H)
      I1=II-1
      J1=JJ-1
      H2=H*H
      DO 1 I=2,I1
```

```
      IR=IRHS(I)
      IO=IST(I)
      IM=IST(I-1)
      IP=IST(I+1)
      DO 1 J=2,J1
      A=-Q(IR+J)-Q(IO+J+1)-Q(IO+J-1)-Q(IM+J)-Q(IP+J)
    1 Q(IR+J)=-A-4.*Q(IO+J)
      RETURN
      END
C
      SUBROUTINE RELAX(K,KRHS,WU,M,ERRM)
c-----Gauss-Seidel
      implicit real*8 (a-h,o-z)
      COMMON Q(18000)
      DIMENSION IST(200),IRHS(200)
      CALL KEY(K,IST,II,JJ,H)
      CALL KEY(KRHS,IRHS,II,JJ,H)
      I1=II-1
      J1=JJ-1
      ERR=0.
      ERRQ=0.
      ERRM=0.
      H2=H*H
      COEFF=4.

      DO 1 I=2,I1
      IR=IRHS(I)
      IQ=IST(I)
      IM=IST(I-1)
      IP=IST(I+1)
      DO 1 J=2,J1
      A=Q(IR+J)-Q(IQ+J+1)-Q(IQ+J-1)-Q(IM+J)-Q(IP+J)
c-----residual norm L2
      ERR=ERR+(A+COEFF*Q(IQ+J))**2
      QOLD=Q(IQ+J)
      Q(IQ+J)=-A/(COEFF)
      ERRQ=ERRQ+(QOLD-Q(IQ+J))**2
c-----relative 'dynamic' error norm max
      Z=abs(QOLD-Q(IQ+J))
      ERRM=MAX(ERRM,Z)
    1 CONTINUE
```

```
      ERR=SQRT(ERR)/H
      ERRQ=SQRT(ERRQ)
      WU=WU+4.**(K-M)
      write(*,2) K,ERRQ,WU
    2 FORMAT('  LEVEL',I2,' RESIDUAL NORM=',E10.3,'   WORK=',F7.3)
      RETURN
      END
C
```

8 Appendix: A 2D MG code for an optimality system in MATLAB

The following is a MATLAB multilevel algorithm for the solution of a linear optimality system.

This section has been contributed by Ian Kopacka.

8.1 Problem Definition

Let us consider the following PDE constrained optimal control problem with tracking-type cost functional:

$$\begin{aligned} \min J(y,u) &:= \tfrac{1}{2}\|y-z\|_{L^2}^2 + \tfrac{\nu}{2}\|u\|_{L^2}^2 \\ \text{s.t.} \quad \Delta y &= u+f \quad \text{in } \Omega \\ y &= 0 \qquad \text{on } \partial\Omega \end{aligned} \tag{109}$$

where Ω is an open, bounded subset of $\mathbb{R}^n$ with a piecewise Lipschitz continuous boundary, ν is a positive constant and $z,\ f \in L^2(\Omega)$. It is well known that the system above has a unique solution. Object of this report is solving problem 109 numerically on the one dimensional unit interval $(0,1)$ as well as on the two dimensional unit square $(0,1)\times(0,1)$ using a multilevel method.

8.1.1 Optimality conditions

In order to derive the first order optimality conditions for problem 109 we define the Lagrange functional

$$\mathcal{L}(y,u,p) := J(y,u) + \langle \Delta y - u - f, p\rangle_{H^{-1},H_0^1}.$$

The necessary optimality conditions are then given by

$$\begin{aligned} \nabla_{(y,u)}\mathcal{L}(y,u,p) &= 0 \\ \Delta y &= u + f \quad \text{in } \Omega \\ y &= 0 \qquad \text{on } \partial\Omega \end{aligned} \tag{110}$$

Derivation with respect to the state variable y yields the adjoint equation, derivation with respect to the control variable u yields the optimality condition. Together with the state equation they form the optimality system:

$$\Delta y - u = f \quad (state\ equation) \tag{111a}$$
$$\Delta p + y = z \quad (adjoint\ equation) \tag{111b}$$
$$\nu u - p = 0 \quad (optimality\ condition) \tag{111c}$$

8.2 Discretization

The domain Ω is discretized using equidistant grids with step length h_k on the k-th level. The grids are coarsened by doubling the step length i.e. $h_{k-1} = 2h_k$, where $k_{min} \leq k \leq k_{max}$.

8.2.1 One dimensional case

The unit interval $\Omega := (0,1)$ is discretized using a step length $h_k := 2^{-k}$ for $k \in \mathbb{N}$, $k > 0$. The inner grid points are defined by $x^j := j \cdot h_k$ for $j = 1, \ldots, n_k$, where $n_k := 2^k - 1$. The Laplace-Operator is discretized using the standard three point-finite difference discretization. Interpolation between two grids is done using linear interpolation. The restriction is done using full weighting with the stencil $\frac{1}{4}[1\ 2\ 1]$.

8.2.2 Two dimensional case

The unit square $\Omega := (0,1)\times(0,1)$ is discretized using the same step length $h_k := 2^{-k}$ for $k \in \mathbb{N}$, $k > 0$ in each direction. The inner grid points are defined by $((x_1^i, x_2^j)) := (i \cdot h_k, j \cdot h_k)$ for $i, j = 1, \ldots, n_k$, where $n_k := 2^k - 1$. The Laplace-Operator is discretized using the standard five point-finite difference star. Interpolation between two grids is done using linear interpolation. The restriction is done using full weighting with the stencil $\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$.

8.3 Algorithm

Problem 109 is solved using a **Full Approximation Scheme (FAS)**. In the following Algorithm I_k^{k-1} denotes the full-weighting restriction operator, $\hat{I}_k^{k-1}$ denotes the straight-injection restriction operator and I_{k-1}^k denotes the linear interpolation operator.

Algorithm 11 FAS

1. *Choose* $k_{max} > k_{min} \in \mathbb{N}$, y^0, u^0, p^0 *on the finest grid,* ν_{pre}, $\nu_{post}, \gamma \in \mathbb{N}$. *Set* $\omega^0 := (y^0, u^0, p^0)$, $k := k_{max}$.

2. *If* $k = k_{min}$ *solve system directly.*

3. *Execute* ν_{pre} *pre-smoothing steps on the fine grid:* $\omega_k^l := S(\omega_k^{l-1}, f_k, z_k)$ *for* $l = 1, \ldots, \nu_{pre}$.

4. *Compute the residual of the state and the adjoint equation:* $r_{s,k} := f_k - \Delta_k y_k^{\nu_{pre}} + u_k^{\nu_{pre}}$, $r_{a,k} := z_k - \Delta_k p_k^{\nu_{pre}} - y_k^{\nu_{pre}}$.

5. *Restrict the residual to a coarser grid:* $r_{s,k-1} := I_k^{k-1} r_{s,k}$, $r_{a,k-1} := I_k^{k-1} r_{a,k}$.

6. *Set* $f_{k-1} := r_{s,k-1} + \Delta_{k-1}\left(\hat{I}_k^{k-1} y_k\right) - \hat{I}_k^{k-1} u_k$, $z_{k-1} := r_{a,k-1} + \Delta_{k-1}\left(\hat{I}_k^{k-1} p_k\right) + \hat{I}_k^{k-1} y_k$.

7. *Call* γ *times FAS to solve*

$$\begin{cases} \Delta_{k-1} y_{k-1} - u_{k-1} &= f_{k-1} \\ \Delta_{k-1} p_{k-1} + y_{k-1} &= z_{k-1} \\ \nu u_{k-1} - p_{k-1} &= 0 \end{cases}$$

8. *Coarse grid correction: Set* $y_k^{\nu_{pre}+1} := y_k^{\nu_{pre}} + I_{k-1}^k \left(y_{k-1} - \hat{I}_k^{k-1} y_k^{\nu_{pre}}\right)$, $p_k^{\nu_{pre}+1} := p_k^{\nu_{pre}} + I_{k-1}^k \left(p_{k-1} - \hat{I}_k^{k-1} p_k^{\nu_{pre}}\right)$, $u_k^{\nu_{pre}+1} := \frac{1}{\nu} p_k^{\nu_{pre}+1}$.

9. *Execute* ν_{post} *post-smoothing steps on the fine grid:* $\omega_k^l := S(\omega_k^{l-1}, f_k, z_k)$ *for* $l = \nu_{pre} + 2, \ldots, \nu_{pre} + \nu_{post} + 1$.

8.4 Smoothing

Smoothing is performed using a collective Gauss-Seidel type scheme, where the equation is solved for each point x_i or (x_1^i, x_2^j) respectively, fixing all other points and looping over all indices $i \in \{1, \ldots, n\}$ or $(i, j) \in \{1, \ldots, n\}^2$ respectively. In the one

dimensional case the following smoothing is done successively for each index i. Here y_i denotes $y(x^i)$, p, u, f and z are treated analogously:

$$\begin{aligned} A_i &:= y_{i+1} + y_{i-1} - h^2 f_i, \\ B_i &:= p_{i+1} + p_{i-1} - h^2 z_i, \\ u_i &= \tfrac{1}{4\nu+h^4}(2B_i + h^2 A_i), \\ y_i &= \tfrac{1}{2}(A_i - h^2 u_i), \\ p_i &= \nu u_i. \end{aligned}$$

In the two dimensional case the smoothing is computed as follows. $y_{i,j}$ denotes $y(x_1^i, x_2^j)$, p, u, f and z are treated analogously:

$$\begin{aligned} A_{i,j} &:= y_{i+1,j} + y_{i-1,j} + y_{i,j+1} + y_{i,j-1} - h^2 f_{i,j}, \\ B_{i,j} &:= p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} - h^2 z_{i,j}, \\ u_{i,j} &= \tfrac{1}{16\nu+h^4}(4B_{i,j} + h^2 A_{i,j}), \\ y_{i,j} &= \tfrac{1}{4}(A_{i,j} - h^2 u_{i,j}), \\ p_{i,j} &= \nu u_{i,j}. \end{aligned}$$

8.5 Numerical results

All computations were done on a Pentium D, 3GHz personal computer using MATLAB version 7.1.0.246 (R14), Service Pack 3.

8.5.1 Test problem 1

We consider the one dimensional domain $\Omega = (0, 1)$. We define f, $z \in L^2(\Omega)$ by

$$\begin{aligned} f(x) &:= -4\pi^2 \sin(2\pi x) - x(x-1), \\ z(x) &:= 2\nu + \sin(2\pi x). \end{aligned}$$

The exact solution is then given by:

$$\begin{aligned} y^*(x) &= \sin(2\pi x), \\ u^*(x) &= x(x-1), \\ p^*(x) &= \nu x(x-1). \end{aligned}$$

The parameters are $\nu = 1e{-}3$, $\gamma = 1$, $\nu_{pre} = \nu_{post} = 2$. The algorithm is terminated, as soon as the relative residuals $\text{res}_s := \|\Delta_h y - u - f\|/\|f\|$ and $\text{res}_a := \|\Delta_h p + y - z\|/\|z\|$ fall below a tolerance ε. The tolerance is fixed with $\varepsilon = 1e{-}6$. The algorithm is initialized with the oscillating functions $y_0(x) := u_0(x) := p_0(x) := \sin(20\pi x)$.

Table 4: Results for test problem 1 with $k_{min} = 2$.

k_{max}	iter	time (s)
14	8	0.2500
15	8	0.4531
16	8	1.1406
17	8	2.4688
18	8	5.3438

The results in table 4 show the independence of the FAS on the grid, as well as linear dependence of the computational cost on the number of grid points. Increasing k_{max} by one means roughly doubling the number of inner grid points. It can be observed that the CPU time doubles as well.

Figure 9 shows convergence results for the finest grid $k_{max} = 18$. It suggests superlinear convergence of the state variable y and linear convergence of the control u.

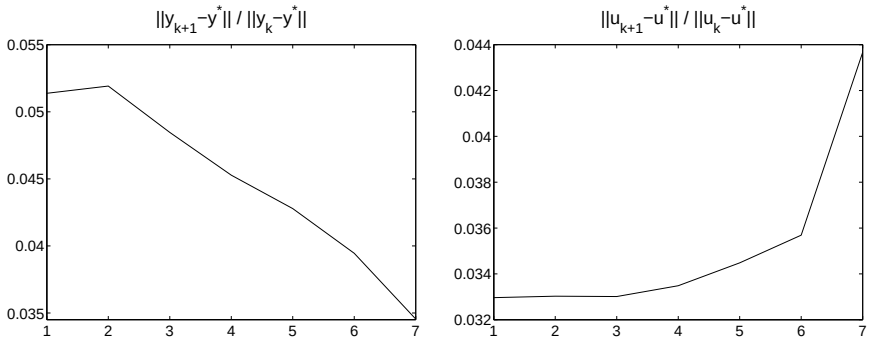

Figure 9: Convergence of state and control in test problem 1.

8.6 Test problem 2

Once again we consider the one dimensional domain $\Omega = (0, 1)$. We define f, $z \in L^2(\Omega)$ by

$$\begin{aligned} f(x) &:= \begin{cases} 1 & \text{on } (0.25, 0.75) \\ 0 & \text{else} \end{cases}, \\ z(x) &:= \max(0, 1 - 10(x - 0.5)^2). \end{aligned}$$

The parameters are $\nu = 1e-3$, $\gamma = 1$, $\nu_{pre} = \nu_{post} = 2$. The same stopping criterion and initialization is used as in the previous test problem.

Table 5: Results for test problem 2 with $k_{min} = 2$.

k_{max}	iter	time (s)
10	8	0.3750
11	8	0.6875
12	8	1.2813
13	8	2.5000
14	8	4.9531
15	8	9.9375
16	8	20.0938

The results in table 5 show the independence of the FAS on the grid, as well as linear dependence of the computational cost on the number of grid points.

8.6.1 Test problem 3

We consider the two dimensional domain $\Omega = (0,1) \times (0,1)$. We define $f,\ z \in L^2(\Omega)$ by

$$\begin{aligned} f(x_1,x_2) &:= -4\pi^2 \sin(2\pi x_1)(2\cos(2\pi x_2) - 1) - \sin(\pi x_1)x_2(x_2-1), \\ z(x_1,x_2) &:= \nu \sin(\pi x_1)(2 - \pi^2 x_2(x_2-1)) + \sin(2\pi x_1)(\cos(2\pi x_2) - 1). \end{aligned}$$

The exact solution is then given by:

$$\begin{aligned} y^*(x_1,x_2) &= \sin(2\pi x_1)(\cos(2\pi x_2) - 1), \\ u^*(x_1,x_2) &= \sin(\pi x_1)x_2(x_2-1), \\ p^*(x_1,x_2) &= \nu \sin(\pi x_1)x_2(x_2-1). \end{aligned}$$

The parameters are $\nu = 1e-3$, $\gamma = 1$, $\nu_{pre} = \nu_{post} = 2$. Stopping criterion for the algorithm is the same as in the one dimensional case. The algorithm is initialized with the oscillating functions $y_0(x_1,x_2) := u_0(x_1,x_2) := p_0(x_1,x_2) := \sin(20\pi x_1)(\cos(20\pi x_2) - 1)$.

The results in table 6 again show the independence of the FAS on the grid, as well as linear dependence of the computational cost on the number of grid points. Increasing k_{max} by one means roughly multiplying the number of inner grid points by a factor 4. It can be observed that the CPU time are also approximately multiplied by 4.

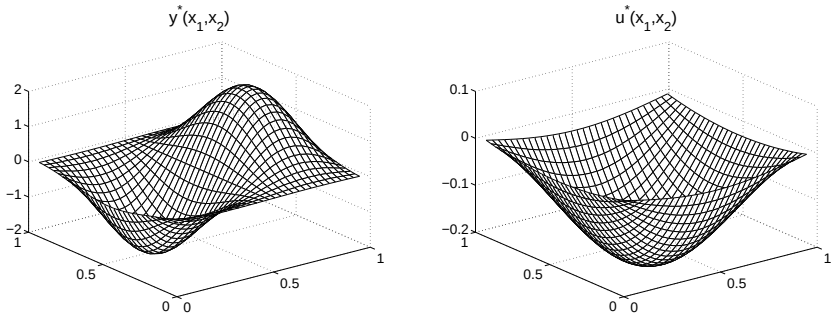

Figure 10: Exact solutions y^* and u^* for test problem 3.

Table 6: Results for test problem 3 with $k_{min} = 2$.

k_{max}	iter	time (s)
6	9	0.1094
7	9	0.2656
8	10	1.5625
9	10	7.1406
10	10	30.4688
11	10	128.0781

8.6.2 Test problem 4

We consider the two dimensional domain $\Omega = (0,1)\times(0,1)$. We define $f,\ z \in L^2(\Omega)$ by

$$\begin{aligned} f(x_1,x_2) &:= \begin{cases} 1 & \text{on } (0.25,0.75)\times(0.25,0.75) \\ 0 & \text{else} \end{cases}, \\ z(x_1,x_2) &:= \max(0,1-10(x_1-0.5)^2)\max(0,1-10(x_2-0.5)^2). \end{aligned}$$

The parameters are $\nu = 1e-3$, $\gamma = 1$, $\nu_{pre} = \nu_{post} = 2$. The same stopping criterion and initialization is used as in the previous test problem.
As before the results displayed in table 7 verify grid independence and computational cost of $\mathcal{O}(n)$.

8.6.3 Conclusion

In all test examples we could verify the grid independence of the FAS scheme and the computational cost of $\mathcal{O}(n)$. In general we only had linear convergence of the discrete L^2 norm of the residuals and the functions themselves. Comparing the CPU

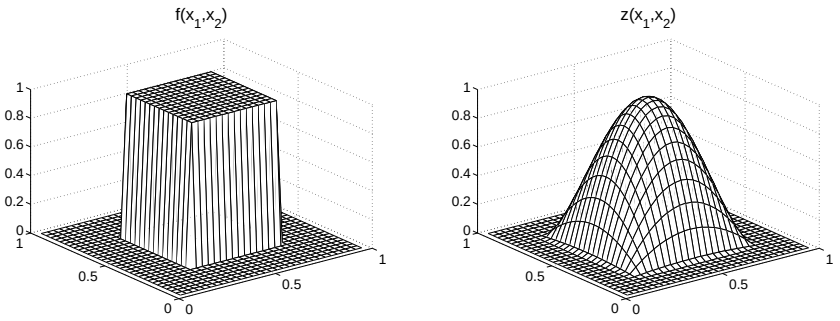

Figure 11: f and z for test problem 4.

Table 7: Results for test problem 4 with $k_{min} = 2$.

k_{max}	iter	time (s)
4	8	0.1875
5	8	0.6250
6	9	2.3594
7	9	8.8906
8	10	39.1719
9	10	159.5625

times we can conclude that the problems with non-differentiable right hand sides f and z, i.e. problems 2 and 4 were significantly harder to solve than the "smooth" problems 1 and 3.

8.7 MATLAB code

In this section we present the MATLAB code solving the two dimensional test problem 3 using FAS.

```
% MAIN PROGRAM
% multi-level scheme solving the optimal control problem
%
%           min  1/2*|y-z|^2 + nu/2*|u|^2
%           s.t. laplace y = u + f
%
% on the 2D domain (0,1)x(0,1).

clear fprintf('\n\nMULTILEVEL ALGORITHM:\n') fprintf('build data
```

```
...')

% grid:
% -----
% the domain (0,1)x(0,1) is discretized using an equidistant grid
% with (2^k-1) inner grid points per dimension on each level of
% the cycle. the mesh size is given by h = 1/(2^k). the finest
% grid is defined by k_max, the coarsest by k_min.
% the mesh sizes of the grids are therefore defined by:
%           h = [1/(2^k_max),1/(2^(k_max-1)),...,1/(2^k_min)]
k_max = 10;     % finest mesh size: 1/(2^k_max)
k_min = 2;      % coarsest mesh size: 1/(2^k_min)

% parameters:
% -----------
gamma = 1;         % number of recursive multilevel calls
pre = 2;           % number of pre-smoothing steps
post = 2;          % number of post-smoothing steps
tolerance = 1e-9;  % tolerance for stopping criterion
counter_max = 20;  % maximal number of cycles

% initialize values:
% ------------------
% build finest grid:
h = 2^(-k_max);    % mesh size
h_inv = 1/h; grid = linspace(h,1-h,2^k_max -1)'; [xgrid,ygrid] =
meshgrid(grid);
% weighting parameter in cost functional:
nu = 1e-3;
% right-hand sides:
f = -4*pi*pi*sin(2*pi*xgrid).*(2*cos(2*pi*ygrid) - 1) - ...
    sin(pi*xgrid).*(ygrid.*ygrid - ygrid);
z = nu*sin(pi*xgrid).*(2 - pi*pi*(ygrid.*ygrid - ygrid)) +...
    sin(2*pi*xgrid).*(cos(2*pi*ygrid) - 1);
% norms of right hand sides (for relative residual):
norm_f_inv = 1/norm(f,'fro'); norm_z_inv = 1/norm(z,'fro');

% exact solutions:
y_exact = sin(2*pi*xgrid).*(cos(2*pi*ygrid)-1); u_exact =
sin(pi*xgrid).*(ygrid.*ygrid - ygrid); p_exact = nu*u_exact;
% initial guess:
```

```
y = sin(20*pi*xgrid).*(cos(20*pi*ygrid)-1); p =
sin(20*pi*xgrid).*(cos(20*pi*ygrid)-1); u =
sin(20*pi*xgrid).*(cos(20*pi*ygrid)-1);
% initialize residual:
rel_resid_state = norm(neg_lap02(y,h_inv*h_inv) + f + u,...
    'fro')*norm_f_inv;
rel_resid_adj   = norm(neg_lap02(p,h_inv*h_inv) + z - y,...
    'fro')*norm_z_inv;
% initialize counter for cycles:
counter = 0; fprintf('done!\n')

% start multilevel scheme:
% ------------------------
fprintf('start mutilevel scheme ...\n') start_time = cputime; while
((rel_resid_state > tolerance) | (rel_resid_adj > tolerance) &...
        (counter <= counter_max))
   % call recursive multilevel scheme:
   [y,p,u] = multilevel_recursive_oc01(y,p,u,f,z,nu,gamma,...
       pre,post,k_max,k_min);
   % increase counter:
   counter = counter + 1;
   % compute relative residuals:
   rel_resid_state = norm(neg_lap02(y,h_inv*h_inv) + f + u,...
       'fro')*norm_f_inv;
   rel_resid_adj   = norm(neg_lap02(p,h_inv*h_inv) + z - y,...
       'fro')*norm_z_inv;
   fprintf('   iter %4i, relres_state = %6.4e, relres_adj = %6.4e\n',...
       counter,rel_resid_state,rel_resid_adj);
   fprintf('   |y - y_ex| = %6.4e\n',norm(y-y_exact,'fro')*h*h);
   fprintf('   |u - u_ex| = %6.4e\n',norm(u-u_exact,'fro')*h*h);
end fprintf('done!\n') elapsed_time = cputime - start_time;

% output:
fprintf('results:\n')
fprintf('  no. of cycles: %4i\n',counter)
fprintf('  relative residual of state eq.: %6.4e\n',rel_resid_state)
fprintf('  relative residual of adj. eq. : %6.4e\n',rel_resid_adj)
fprintf('  elapsed time (s): %8.4f\n',elapsed_time)

fprintf('data:\n')
fprintf('  no. of inner grid points on finest grid:   %6i\n',...
```

```
    (2^k_max-1)^2)
fprintf('  no. of inner grid points on coarsest grid: %6i\n',...
    (2^k_min-1)^2)
fprintf('  k_min, k_max : %2i, %2i\n', k_min,k_max)
fprintf('  gamma = %2i, nu1 = %2i, nu2 = %2i\n',gamma,pre,post)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% RECURSIVE MULTILEVEL SOLVER
%
%  [y,p,u] = multilevel_recursive_oc01(y0,p0,u0,f,z,nu,gamma,pre,post,k,k_min)
%
% input parameters:
%     y0......initial guess for state
%     p0......initial guess for lagrangean multiplier
%     u0......initial guess for control
%     f.......right hand side of state equation
%     z.......desired state
%     nu......weighting parameter in cost functional
%     gamma...number of recursive calls
%     pre.....number of pre-smoothing steps
%     post....number of post-smoothing steps
%     k.......number corresponding to grid (h = 1/2^k)
%     k_min...coarsest grid
% output parameter:
%     y.......state
%     p.......lagrangean multiplier
%     u.......control
function [y,p,u] = ...
    multilevel_recursive_oc01(y0,p0,u0,f,z,nu,gamma,pre,post,k,k_min)

% mesh size:
h_inv = 2^k; h = 1/h_inv;

% if on coarsest grid, then solve exactly:
if k <= k_min
    % build matrix:
    n_temp = 2^k-1;
    e = ones(n_temp,1)*h_inv;
```

```
    A = spdiags([e -2*e e],-1:1,n_temp,n_temp);
    % 2D (positive) Laplace-matrix A:
    I_h = speye(n_temp)*h_inv;
    A = kron(I_h,A) + kron(A,I_h);    % n2 x n2
    clear e I_h

    % solve exactly:
    yp = [A,-1/nu.*speye(n_temp*n_temp);speye(n_temp*n_temp),A]\...
        [f(:);z(:)];
    y = reshape(yp(1:n_temp*n_temp),n_temp,n_temp);
    p = reshape(yp(n_temp*n_temp+1:end),n_temp,n_temp);
    u = p./nu;
    return

    % otherwise smooth, compute residual, restrict, call recursive
    % multilevel scheme, compute coarse-grid-correction and smooth
    % again:
else
    % initialize solution:
    y = y0;
    p = p0;
    u = u0;

    % perform multigrid scheme gamma times:
    for rv = 1:gamma
        % coarse mesh size:
        H_inv = 2^(k-1);
        H = 1/H_inv;
        % pre-smoothing:
        [y,p,u] = smooth_oc01(y,p,u,f,z,nu,pre);

        % compute residual:
        resid_state = f + neg_lap02(y,h_inv*h_inv) + u;
        resid_adj   = z + neg_lap02(p,h_inv*h_inv) - y;

        % restrict to coarser grid:
        resid_state_coarse = restrict02(resid_state);
        resid_adj_coarse = restrict02(resid_adj);

        % compute straight injection:
        y_strinj_coarse = restrict_strinj_2D(y);
```

```
        p_strinj_coarse = restrict_strinj_2D(p);
        u_strinj_coarse = restrict_strinj_2D(u);


        % right hand side of coarse problem:
        f_coarse = resid_state_coarse - ...
            neg_lap02(y_strinj_coarse,H_inv*H_inv) - u_strinj_coarse;
        z_coarse = resid_adj_coarse - neg_lap02(p_strinj_coarse,...
            H_inv*H_inv) + y_strinj_coarse;

        % apply recursive multilevel scheme:
        [y_coarse,p_coarse,u_coarse] = ...
        multilevel_recursive_oc01(y_strinj_coarse,...
        p_strinj_coarse,u_strinj_coarse,f_coarse,z_coarse,...
        nu,gamma,pre,post,k-1,k_min);


        % coarse-grid-correction:
        y = y + interpolate02(y_coarse - y_strinj_coarse);
        p = p + interpolate02(p_coarse - p_strinj_coarse);
        u = 1/nu.*p;

        % post-smoothing:
        [y,p,u] = smooth_oc01(y,p,u,f,z,nu,post);
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% COLLECTIVE GAUSS-SEIDEL SMOOTHING
%
% [y,p,u] = smooth_oc01(y,p,u,f,z,nu,cycles)
%
% input parameters:
%    y........state
%    p........lagrange multiplier
%    u........control
%    f........right hand side of state equation
%    z........desired state
```

```
%    nu.......weighting parameter in cost functional
%    cycles...number of smoothing cycles
% output parameter:
%    y........state after smoothing
%    p........multiplier after smoothing
%    u........control after smoothing
function [y,p,u] = smooth_oc01(y,p,u,f,z,nu,cycles)

% mesh size h:
[n,m] = size(f); h = 1/(n + 1); h2 = h*h;

% embed variables in zeros:
y = [zeros(1,n+2); zeros(n,1),y,zeros(n,1); zeros(1,n+2)]; p =
[zeros(1,n+2); zeros(n,1),p,zeros(n,1); zeros(1,n+2)];

% loop over number of cycles:
for rv_cycles = 1:cycles
    % running variable over columns:
    for rvc = 2:n+1
        % running variable over rows:
        for rvr = 2:n+1
            Aij = y(rvr+1,rvc) + y(rvr-1,rvc) + y(rvr,rvc+1) + ...
                y(rvr,rvc-1) - h2*f(rvr-1,rvc-1);
            Bij = p(rvr+1,rvc) + p(rvr-1,rvc) + p(rvr,rvc+1) + ...
                p(rvr,rvc-1) - h2*z(rvr-1,rvc-1);
            u(rvr-1,rvc-1) = (4*Bij + h2*Aij)/(16*nu + h2*h2);
            y(rvr,rvc) = (Aij - h2*u(rvr-1,rvc-1))/4;
            p(rvr,rvc) = nu*u(rvr-1,rvc-1);
        end
    end
end

% eliminate the zero boundary values:
y = y(2:n+1,2:n+1); p = p(2:n+1,2:n+1);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% 2D LINEAR INTERPOLATION OPERATOR
%
```

```
% fine = interpolate02(coarse)
%
% input parameters:
%    coarse......matrix of dimension (2^(k-1)-1)x(2^(k-1)-1)
% output parameter:
%    fine........matrix of dimension (2^k-1)x(2^k-1)
function fine = interpolate02(coarse)

% initialize vector:
[m,n] = size(coarse); fine  = zeros(2*m + 1,2*m + 1);

% set on coarse grid:
fine(2:2:end-1,2:2:end-1) = coarse;
% interpolate in x-direction:
fine(2:2:end-1,1:2:end-2) = 0.5.*coarse; fine(2:2:end-1,3:2:end) =
fine(2:2:end-1,3:2:end) + 0.5.*coarse;

% interpolate in y-direction:
fine(1:2:end-2,:) = 0.5.*fine(2:2:end-1,:); fine(3:2:end,:) =
fine(3:2:end,:) + 0.5.*fine(2:2:end-1,:);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% 2D FULL WEIGHTING RESTRICTION OPERATOR
%
% coarse = restrict02(fine)
%
% input parameters:
%    fine........matrix of dimension (2^k-1)x(2^k-1)
% output parameter:
%    coarse......matrix of dimension (2^(k-1)-1)x(2^(k-1)-1)
function coarse = restrict02(fine)

coarse = fine(2:2:end-1,2:2:end-1)./4 + ...
    (fine(1:2:end-2,2:2:end-1) + fine(3:2:end,2:2:end-1) + ...
    fine(2:2:end-1,1:2:end-2) + fine(2:2:end-1,3:2:end))./8 + ...
    (fine(1:2:end-2,1:2:end-2) + fine(3:2:end,3:2:end) + ...
    fine(1:2:end-2,3:2:end) + fine(3:2:end,1:2:end-2))./16;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 2D STRAIGHT INJECTION RESTRICTION OPERATOR
%
% coarse = restrict_strinj_2D(fine)
%
% input parameters:
%    fine........matrix of dimension (2^k-1) x (2^k-1)
% output parameter:
%    coarse......matrix of dimension (2^(k-1)-1) x (2^(k-1)-1)
function coarse = restrict_strinj_2D(fine)

[n,m] = size(fine); coarse = fine(2:2:n-1,2:2:m-1);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 2D FIVE POINT NEGATIVE LAPLACE OPERATOR
%
% v = neg_lap02(u,h_inv2)
%
% input parameters:
%    u........matrix
%    h_inv2...1/h^2, where h is the mesh size in both directions
% output parameter:
%    v........matrix -Delta u, same size as u
function v = neg_lap02(u,h_inv2)

% compute negative laplacean:
vx = [2*u(:,1) - u(:,2), -u(:,1:end-2) + 2*u(:,2:end-1) - ...
    u(:,3:end), -u(:,end-1) + 2*u(:,end)];
vy = [2*u(1,:) - u(2,:); -u(1:end-2,:) + 2*u(2:end-1,:) - ...
    u(3:end,:); -u(end-1,:) + 2*u(end,:)];

v = (vx + vy)*h_inv2;
```

References

[1] N.S. Bakhvalov, *On the convergence of a relaxation method with natural constraints on the elliptic operator*, *USSR Computational Math. and Math. Phys.*, 6 (1966), pp. 101.

[2] A. Borzì, *On the convergence of the MG/OPT method*, PAMM, Proceedings GAMM Annual Meeting - Luxembourg 2005, 5(1)(2005), pp. 735–736.

[3] A. Borzì, *Space-time multigrid methods for solving unsteady optimal control problems*, to appear in the Proceedings of the Second Sandia Workshop on PDE-Constrained Optimization: Toward Real-time and Online PDE-constrained Optimization, May 19-21, 2004, Bishop's Lodge, Santa Fe, New Mexico. Workshop organizing committee: Larry Biegler, Omar Ghattas, Matthias Heinkenschloss, David Keyes, Bart van Bloemen Waanders.

[4] A. Borzì, *High-order discretization and multigrid solution of elliptic nonlinear constrained optimal control problems*, J. Comp. Appl. Math., 200 (2007), pp. 67–85.

[5] A. Borzì, *Multigrid methods for parabolic distributed optimal control problems*, J. Comp. Appl. Math., 157 (2003), pp. 365–382.

[6] A. Borzì and G. Borzì, *An algebraic multigrid method for a class of elliptic differential systems*, SIAM J. Sci. Comp., 25(1) (2003), pp. 302–323.

[7] A. Borzì and G. Borzì, *An efficient algebraic multigrid method for solving optimality systems*, Computing and Visualization in Science, 7(3/4) (2004), pp. 183–188.

[8] A. Borzì and K. Kunisch, *The numerical solution of the steady state solid fuel ignition model and its optimal control*, SIAM J. Sci. Comp., 22(1) (2000), pp. 263–284.

[9] A. Borzì and K. Kunisch, *A globalization strategy for the multigrid solution of elliptic optimal control problems*, Optimization Methods and Software, 21(3) (2006), 445-459.

[10] A. Borzì, K. Kunisch, and D.Y. Kwak, *Accuracy and convergence properties of the finite difference multigrid solution of an optimal control optimality system*, SIAM J. Control Opt., 41(5) (2003), pp. 1477-1497.

[11] J.H. Bramble, *Multigrid Methods*, Pitman research notes in mathematical series, Longman Scientific & Technical, 1993.

[12] J.H. Bramble, J.E. Pasciak, and J. Xu, *The analysis of multigrid algorithms with nonnested spaces or noninherited quadratic forms*, Math. Comp., 56 (1991), pp. 1–34.

[13] A. Brandt, *Multi-Level Adaptive Technique (MLAT) for Fast Numerical Solution to Boundary-Value Problems*. In: H. Cabannes and R. Temam (eds.), Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics, Paris 1972. *Lecture Notes in Physics* 18, Springer-Verlag, Berlin, 1973.

[14] A. Brandt, *Multi-Level Adaptive Solutions to Boundary-Value Problems*, *Math. Comp.*, 31 (1977), pp. 333-390.

[15] A. Brandt, S. McCormick and J. Ruge, *Multigrid Methods for Differential Eigenproblems*, *SIAM J. Sci. Stat. Comput.*, 4 (1983), pp. 244-260.

[16] A. Brandt, *Multi-grid techniques: 1984 guide with applications to fluid dynamics*. GMD-Studien. no 85, St. Augustin, Germany, 1984.

[17] A. Brandt, *Rigorous Local Mode Analysis of Multigrid*. Lecture at the 2nd European Conference on Multigrid Methods, Cologne, Oct. 1985. Research Report, The Weizmann Institute of Science, Israel, Dec. 1987.

[18] A. Brandt and J. Greenwald, *Parabolic Multigrid Revisited*. In: Hackbusch-Trottenberg [32].

[19] A. Brandt and D. Sidilkover , *Multigrid solution to steady-state two-dimensional conservation laws*, SIAM Journal on Numerical Analysis, 30 (1993), pp. 249-274.

[20] Th. Dreyer, B. Maar, V. Schulz, Multigrid optimization in applications, J. Comput. Appl. Math., 120 (2000), pp. 67–84.

[21] R.P. Fedorenko, *A relaxation method for solving elliptic difference equations*. *USSR Computational Math. and Math. Phys.*, 1 (1962), pp. 1092.

[22] R.P. Fedorenko, *The rate of convergence of an iterative process*. *USSR Computational Math. and Math. Phys.*, 4 (1964), pp. 227.

[23] J. Goodman and A.D. Sokal, *Multigrid Monte Carlo for lattice field theories*, *Phys. Rev. Lett.*, 56 (1986), pp. 1015.

[24] W. Hackbusch, *A multi-grid method applied to a boundary problem with variable coefficients in a rectangle*. Report 77-17, Institut für Angewandte Mathematik, Universität Köln, 1977.

[25] W. Hackbusch, *On the computation of approximate eigenvalues and eigenfunctions of elliptic operators by means of a multi-grid method*, *SIAM J. Numer. Anal.*, 16 (1979), pp. 201-215.

[26] W. Hackbusch, *Parabolic Multi-Grid Methods*. In: R. Glowinski and J.L. Lions (eds.), *Computing Methods in Applied Sciences and Engineering*, VI. Proc. of the sixth international symposium, Versailles, Dec.1983, North-Holland, Amsterdam, 1984.

[27] W. Hackbusch, *Multi-Grid Methods and Applications*. Springer-Verlag, Heidelberg, 1985.

[28] W. Hackbusch, *Elliptic Differential Equations*, Springer-Verlag, New York, 1992.

[29] W. Hackbusch and A. Reusken, Analysis of a damped nonlinear multilevel method, Numer. Math., 55 (1989), pp. 225–246.

[30] W. Hackbusch and U. Trottenberg (eds.), *Multi-Grid Methods*, Proceedings, Köln-Porz, Nov. 1981, *Lecture Notes in Mathematics* 960, Springer-Verlag, Berlin, 1982.

[31] W. Hackbusch and U. Trottenberg (eds.), *Multigrid Methods II*, II. Proc. of the European Conference on Multigrid Methods, Cologne, Oct. 1985, *Lecture Notes in Mathematics* **1228**, Springer-Verlag, Berlin, 1986.

[32] W. Hackbusch and U. Trottenberg (eds.), *Multigrid Methods III*, III. Proc. of the European Conference on Multigrid Methods, Bonn, Oct. 1990, Birkhäuser, Berlin, 1991.

[33] Van Emden Henson, Multigrid for Nonlinear Problems: an Overview, presented by Van Emden Henson at the SPIE 15th Annual Symposium on Electronic Imaging, Santa Clara, California, January 23, 2003.

[34] K. Ito and K. Kunisch, *Asymptotic properties of receding horizon optimal control problems*, SIAM J. Control Optim., 40(5) (2002), pp. 1585–1610.

[35] R.M. Lewis and S.G. Nash, Model problems for the multigrid optimization of systems governed by differential equations, in SIAM Journal on Scientific Computing.

[36] J.L. Lions, *Optimal Control of Systems Governed by Partial Differential Equations*, Springer, Berlin, 1971.

[37] H.D. Mittelmann and H. Weber, *Multi-Grid Methods for Bifurcation Problems*, *SIAM J. Sci. Stat. Comput.*, 6 (1985), pp. 49.

[38] W.A. Mulder, *A new multigrid approach to convection problems*, Journal of Computational Physics, 83 (1989), pp. 303-323.

[39] S.G. Nash, Newton-type minimization via the Lanczos method, SIAM J. Numer. Anal., 21(4) (1984), pp. 770–788.

[40] S.G. Nash, A multigrid approach to discretized optimization problems, Optimization Methods and Software, 14 (2000), pp. 99–116 .

[41] J.W. Ruge and K. Stüben, Algebraic Multigrid (AMG), In "Multigrid Methods" (S. McCormick, ed.), Frontiers in Applied Mathematics, Vol. 5, SIAM, Philadelphia 1986.

[42] P. Spellucci, Numerische Verfahren der nichtlinearen Optimierung, ISNM Birkhäuser, 1993.

[43] K. Stüben, *Algebraic Multigrid (AMG): An Introduction with Applications*, GMD Report 53, March 1999.

[44] K. Stüben and U. Trottenberg, *Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications*. In: Hackbusch-Trottenberg [30].

[45] E. Süli, *Convergence of finite volume schemes for Poisson's equation on nonuniform meshes*, SIAM J. Numer. Anal., 28 (1991), pp. 1419–1430.

[46] S. Ta'asan, *Multigrid Methods for Locating Singularities in Bifurcation Problems*. *SIAM J. Sci. Stat. Comput.*, 11 (1990), pp. 51-62.

[47] S. Ta'asan, *Introduction to shape design and control; Theoretical tools for problem setup; Infinite dimensional preconditioners for optimal design*. In: Inverse Design and Optimisation Methods, VKI LS 1997-05.

[48] X.-C. Tai and J. Xu, Global and uniform convergence of subspace correction methods for some convex optimization problems, Math. Comp., 71 (2002), pp. 105–124.

[49] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.

[50] S. Vandewalle and R. Piessens, *Efficient Parallel Algorithms for Solving Initial-Boundary Value and Time-Periodic Parabolic Partial Differential Equations*, *SIAM J. Sci. Stat. Comput.*, 13 (1992), pp. 1330–1346.

[51] P. Wesseling, *A survey of Fourier smoothing analysis results*. In: Hackbusch-Trottenberg [32].

[52] D. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971. (Reprinted by Dover, 2003.)