BIT Numerical Mathematics (2007) 47: 277–296
DOI: 10.1007/s10543-007-0123-2
Published online: 7 March 2007 – © Springer 2007

AN ACCELERATED ALGEBRAIC MULTIGRID ALGORITHM FOR TOTAL-VARIATION DENOISING*

KE CHEN** and JOSEPH SAVAGE***

[1]*Department of Mathematical Sciences, The University of Liverpool, Peach Street, Liverpool L69 7ZL, UK. e-mail: {k.chen, j.savage}@liverpool.ac.uk*

Abstract.

The variational partial differential equation (PDE) approach for image denoising restoration leads to PDEs with nonlinear and highly non-smooth coefficients. Such PDEs present convergence difficulties for standard multigrid methods. Recent work on algebraic multigrid methods (AMGs) has shown that robustness can be achieved in general but AMGs are well known to be expensive to apply. This paper proposes an accelerated algebraic multigrid algorithm that offers fast speed as well as robustness for image PDEs. Experiments are shown to demonstrate the improvements obtained.

AMS subject classification (2000): 68U10, 65F10, 65K10.

Key words: image restoration, nonlinear partial differential equations, algebraic multigrid methods, acceleration, nonlinear iterations.

1 Introduction.

Image denoising is one of the fundamental problems in image processing and computer vision. During recording and transmission an image will become contaminated with random Gaussian noise; this is modelled by the equation

$$z(x,y) = u(x,y) + n(x,y), \quad x,y \in \Omega \tag{1.1}$$

where Ω is a bounded and open domain of $\mathbb{R}^2$ (usually a rectangle). Here z is a real function representing the observed image, which in practice will only be known at certain discrete values of x and y, u is the true (unknown) image and n is an additive (unknown) Gaussian noise term.

The aim of a denoising process is to remove noise while preserving the main features of the original image, most importantly its edges (an edge is a boundary

* Received January 7, 2005. Accepted in revised form January 7, 2007. Communicated by Per Christian Hansen.
** This work is supported by a research fellowship from the UK Leverhulme Trust RF/9/RFG/2005/0482 (Corresponding author).
*** Support of the UK EPSRC DTA grant is gratefully received.

where a jump in intensity occurs); this can be achieved by the use of Tikhonov regularization with the Total-Variation (TV) regularization functional, first introduced in 1992 by Rudin, Osher and Fatemi [23]:

$$\min_u J(u), \quad J(u) = \int_\Omega \left[\alpha\sqrt{|\nabla u|^2+\beta} + \frac{1}{2}(u-z)^2\right] dxdy, \tag{1.2}$$

where α is a regularization parameter which balances the need for a solution which is a good fit to the observed data and one that is regular, and β is a perturbing parameter which should be small. The resulting Euler–Lagrange equation is

$$u - \alpha\nabla\cdot(D(u)\nabla u) = z \tag{1.3}$$

where $D(u) = 1/\sqrt{|\nabla u|^2+\beta}$, with homogenous Neumann boundary condition $\frac{\partial u}{\partial n} = 0$ at the boundary. Discretization on an $n \times m$ cell-centered Cartesian grid (each cell represents one pixel of the image) Ω^h with cells of size $h \times k$ using a finite difference scheme results in the following equation at grid point (i,j)

$$\begin{aligned} N_h(u)_{ij} &= u_{i,j} - \alpha_h\big[\,(D(u)_{i,j}(u_{i+1,j}-u_{i,j}) - D(u)_{i-1,j}(u_{i,j}-u_{i-1,j})) \\ &\quad + \lambda^2(D(u)_{i,j}(u_{i,j+1}-u_{ij}) - D(u)_{i,j-1}(u_{i,j}-u_{i,j-1}))\big] \\ &= z_{i,j} \end{aligned} \tag{1.4}$$

where

$$D(u)_{i,j} = \big((u_{i+1,j}-u_{ij})^2 + \lambda^2(u_{i,j+1}-u_{ij})^2 + \beta_h\big)^{-1/2} \tag{1.5}$$

and

$$\alpha_h = \alpha/h, \quad \beta_h = h^2\beta \quad \text{and} \quad \lambda = h/k \tag{1.6}$$

with Neumann's boundary conditions

$$u_{i,0} = u_{i,1}, \quad u_{i,m+1} = u_{i,m}, \quad u_{0,j} = u_{1,j}, \quad u_{n+1,j} = u_{n,j}. \tag{1.7}$$

If we stack the grid functions u_h and z_h along rows of Ω^h into vectors $\mathbf{u}$ and $\mathbf{z}$ we see that we can write the system of nonlinear equations in the form

$$A(\mathbf{u})\mathbf{u} = \mathbf{z} \tag{1.8}$$

where $A = I + \alpha_h L(\mathbf{u})\mathbf{u}$ and $L(\mathbf{u})\mathbf{u}$ is the discrete representation of $-\nabla\cdot\big(\frac{\nabla u}{\sqrt{|\nabla u|^2+\beta}}\big)$. Vogel and Oman [31] proposed the following fixed point method to solve this nonlinear equation. Starting from an initial guess $\mathbf{u}^0 = \mathbf{z}$ at step $k+1$ the following linear system is solved to update $\mathbf{u}$

$$A(\mathbf{u}^k)\mathbf{u}^{k+1} = \mathbf{z}. \tag{1.9}$$

This is equivalent to first evaluating the diffusion coefficients $D(u)_{ij}$ at each grid point using the previous iterate and then solving the resulting linear equation. The matrix $A(\mathbf{u}^k)$ is symmetric positive definite and block tridiagonal with positive diagonal entries and negative off-diagonal entries and has rows which sum to 1.

In practice exact solution of the linear Equation (1.9) is not necessary as the nonlinear error is the dominant one and it is usually enough to reduce the residual to a tenth of its original value [10]. In [12] the Krylov acceleration procedure of Washio and Oosterlee [35] was used to accelerate the fixed point method. Throughout this paper we use an inner tolerance of 0.1 in the fixed point method and apply Krylov acceleration with 5 stored solutions to the fixed point iterates.

There exists a large literature of linear iterative methods for solving Equation (1.9); see [29, 31, 30, 32, 4, 5, 3, 21, 18, 27]. However, as pointed out in [2], most iterative solvers are sensitive to the parameter β. More precisely, these methods can be slow to converge for small β because of the highly varying diffusion coefficients. Before we proceed, we must remark that small β is necessary for preserving sharp edges of an image, as stressed by [17, 18, 19] and illustrated for a simple test example below. Figure 1.1 (left) shows a denoising problem of size $N = 128$ (mimicking a line segment of a 2D image) where all four solutions using respectively $\beta = 10^{-3}, 10^{-4}, 10^{-5}, 10^{-8}$ appear acceptable in comparison to the exact solution; however when zooming into one position (pixels 95-111 on the right plot of Figure 1.1) one can see that these large β do not give rise to a solution as sharp as the true TV solution with $\beta = 0$. An alternative method to 'avoid' β is to solve the minimization problem (1.2) by a multigrid method directly as done in [7, 8, 9].

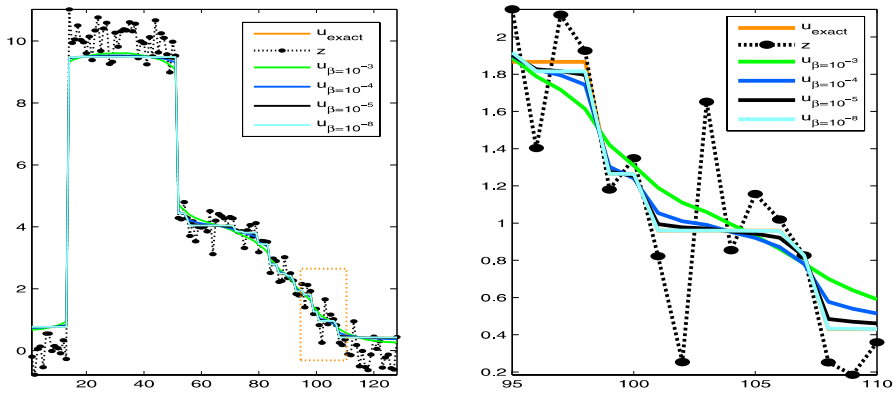

Figure 1.1: Illustration of the necessity of small β for the TV denoising model.

The optimality of multigrid methods suggests that they are the potential solver of choice. For (1.9), in particular, [29, 3] proposed using a linear geometric multigrid method (GMG) to solve the linear equation on each fixed point step. This method was accelerated in [29] by use of the conjugate gradient method. For solving (1.8), in [25, 16], nonlinear multigrid methods were successfully imple-

mented; however large β must be used to avoid large jumps in the coefficients (or to avoid non-convergence). Therefore such methods are not yet robust with respect to β as known from [2]; see also [26] for GMG to other models.

One way to improve the robustness of a GMG with respect to β is by use of algebraic multigrid methods (AMGs) as was done in Chang–Chern [12] who proposed using an AMG for (1.9). As the AMG method defines coarse points and the interpolation operators based on information in the matrix along with adaptive coarse-fine mesh strategy, robustness and fast convergence for solving (1.9) with various β have been achieved. In this short paper, we propose a practical algorithm that attempts to speed up the fixed point with AMG method by recycling AMG setup data.

The rest of the paper is organized as follows. In Section 2, we briefly discuss the AMG method for solving (1.9) with some complexity done in the Appendix. In Section 3 we present our simple idea of accelerating the AMG setup data within the fixed point method, including details of preliminary experiments which motivate the final algorithm given at the end of the section, while in Section 4 we demonstrate how our proposed algorithm improves on the standard fixed point with AMG method. It turns out that our new method significantly speeds up the standard fixed point with AMG method without suffering in robustness.

2 Review of an algebraic multigrid method.

We now give a brief review of an AMG, which could be used as a black box solver for a sparse linear system $A\mathbf{u} = \mathbf{f}$. An AMG should not be considered as a competitor to a GMG, a multigrid method based on a fixed hierarchy of grids, rather it should be used in cases where GMG fails or is too difficult to apply such as elliptic problems with highly varying coefficients or problems based on unstructured grids. It can also be used in problems with no geometric interpretation at all (as implied by the reference of black box solver). Here (1.9) falls into one of these problems with highly varying coefficients.

An AMG was proposed to improve on a GMG: a GMG will not converge unless coarse grids can represent fine grid residuals fairly 'accurately'. This 'accurate' representation is impossible in two related cases. Firstly when the usual smoothers are not effective on an entire fine grid, no standard coarse grids are useful. Then one has to design specific and problem-dependent smoothers in order for a GMG to be applied (which may be difficult). Secondly when the residuals are not uniformly smooth in a fine grid, no coarse grids by standard coarsening are effective. Again one can design specific and problem-dependent coarsening strategies to improve a GMG. An AMG represents an ambitious idea to overcome the above difficulties – the use of non-standard coarsening leads to the reliance on the coefficients of the underlying matrix rather than a geometric grid to provide coarsening and intergrid transfer information. See [12, 13, 34, 28, 15]. Unlike in GMG, the standard smoothers such as point Gauss–Seidel are adequate for AMGs and the coarse levels and transfer operators are defined based on the entries of A. Although convergence has only been established for problems in

which A is symmetric positive definite (SPD), AMGs are robust and have been successfully applied to solve many useful problems.

Given a linear system $A\mathbf{u} = \mathbf{f}$ with $A \in \mathbf{R}^{n\times n}$ and $\mathbf{u} = (u_i)_{i=1,\ldots,n}$, we can consider the index set $\{1,\ldots,n\}$ as a set of points on a fine grid. Denote the neighbours of node i by N_i and the set of points strongly connected to i by S_i. The set of points to which i is strongly connected is denoted by S_i^T as in [12, 28]. AMGs assume that the error in the solution to the linear system can be expected to vary smoothly in the direction of strong connections. An AMG method is made up of 2 phases with the first phase to generate a coarse-fine (nonuniform) mesh splitting and the second phase the application of multigrid cycles.

The *first phase* known as the setup phase is where the multigrid components needed for the second phase are defined. Given the matrix $A^{(1)} = A$ and the index set $\{1,\ldots,n\}$ which we will refer to as $\Omega^{(1)}$ we first split $\{1,\ldots,n\}$ into two disjoint sets C and F where C is the set of coarse points which we will call $\Omega^{(2)}$ and F is the set of fine points. The C/F splitting should attempt to achieve two conditions, the first is that for each point $i \in F$ every point in S_i is either in $C_i = C \cap S_i$ or strongly connected to a point in C_i, the second is that C should be a maximal subset of all points with the property that no two C-points are strongly connected to each other. In practice it is not usually possible to strictly satisfy both conditions, the basic C/F splitting algorithm presented below tries to enforce the second condition by choosing at each step a coarse point with maximal $\lambda_i = |S_i^T \cap U| + 2|S_i^T \cap F|$ where U is the set of points which have yet to be defined as either coarse or fine. We review the main steps of a C/F algorithm with a view to analyze the complexity later (see Appendix):

C/F-Splitting Algorithm. *Set $U = \Omega^{(k)}$, $C = \varnothing$, $F = \varnothing$, $\lambda_i = |S_i^T|$ for all i.*

While $U \neq \varnothing$
  Select $i \in U$ with maximal λ_i.
    $C = C \cup \{i\}$, $U = U - \{i\}$
  For $j \in S_i^T \cap U$
    $F = F \cup \{j\}$, $U = U - \{j\}$
    For $l \in S_j \cap U$
      $\lambda_l = \lambda_l + 1$
    end
  end
  For $j \in S_i \cap U$
    $\lambda_j = \lambda_j - 1$
  end
end

Having obtained this coarse/fine splitting we then define an interpolation operator $I_{(2)}^{(1)}$ for transferring between $\Omega^{(2)}$ and $\Omega^{(1)}$ which will have the general form

$$\left(I_{(2)}^{(1)} e^{(2)}\right)_i = e_i^{(1)} = \begin{cases} e_i^{(2)} & \text{if } i \in C \\ \sum_{k\in P_i} w_{ik} e_k^{(2)} & \text{if } i \in F. \end{cases} \tag{2.1}$$

Here the interpolatory set $P_i \subset C$ is so chosen that it should be reasonably small in order to produce an efficient AMG method; see [24, 13, 28]. We take P_i to be C_i. Once the interpolation operator has been defined, the restriction operator for transferring between $\Omega^{(1)}$ and $\Omega^{(2)}$ is defined to be the transpose of the interpolation operator:

$$I_{(1)}^{(2)} = \big(I_{(2)}^{(1)}\big)^T. \tag{2.2}$$

We then use the Galerkin principle to define the coarse grid matrix $A^{(2)}$:

$$A^{(2)} = I_{(1)}^{(2)}(A^{(1)})I_{(2)}^{(1)}. \tag{2.3}$$

In exactly the same way an even smaller matrix $A^{(3)}$ along with interpolation and restriction operators is defined from the entries of $A^{(2)}$. This process is repeated until we have a sequence of matrices $A^{(1)}, \ldots, A^{(L)}$ with corresponding transfer operators, where $A^{(L)}$ is small enough to be solved efficiently using a direct solver.

In the *second phase*, once the above matrices and transfer operators have been defined and stored it is clearly straightforward to apply a multigrid cycle with point Gauss–Seidel smoother to the original linear problem. Given an initial guess $\mathbf{v}^{(1)}$ and the setup data, one μ-cycle is

$$\mathbf{v}^{(1)} \leftarrow \mathrm{AMG}\mu^{(1)}\big(\mathbf{v}^{(1)}, \mathbf{f}^{(1)}, \nu_1, \nu_2\big) \tag{2.4}$$

where $\mathrm{AMG}\mu^{(k)}$ is defined recursively as follows:

$$\mathbf{v}^{(k)} \leftarrow \mathrm{AMG}\mu^{(k)}\big(\mathbf{v}^{(k)}, \mathbf{f}^{(k)}, \nu_1, \nu_2\big).$$

ALGORITHM 2.1 (The AMG algorithm for (1.9)). *For each step of a fixed point iteration, seek the fast solution of* (1.9) *by the following steps.*

1. *If $A^{(k)}$ is the smallest matrix (on the coarsest level), set $\mathbf{v}^{(k)} = (A^{(k)})^{-1}\mathbf{f}^{(k)}$ and return*
   else do ν_1 steps of smoothing on level k:
   $\mathbf{v}^{(k)} \leftarrow S^{(k)}\mathbf{v}^{(k)} + (I - S^{(k)})(A^{(k)})^{-1}\mathbf{f}^{(k)}$.
2. *Restrict the residual to the coarser level: $\mathbf{f}^{(k+1)} = I_{(k)}^{(k+1)}(\mathbf{f}^{(k)} - A^{(k)}\mathbf{v}^{(k)})$ and set the correction $\mathbf{v}^{(k+1)} = 0$.*
3. *Repeat $\mathbf{v}^{(k+1)} \leftarrow \mathrm{AMG}\mu^{(k+1)}(\mathbf{v}^{(k+1)}, \mathbf{f}^{(k+1)}, \nu_1, \nu_2)$, μ times.*
4. *Add the coarse level correction: $\mathbf{v}^{(k)} \leftarrow \mathbf{v}^{(k)} + I_{(k+1)}^{(k)}\mathbf{v}^{(k+1)}$*
5. *Do ν_2 steps of smoothing on level k: $\mathbf{v}^{(k)} \leftarrow S^{(k)}\mathbf{v}^{(k)} + (I - S^{(k)})(A^{(k)})^{-1}\mathbf{f}^{(k)}$.*

Here $S^{(k)}$ is the Gauss–Seidel smoothing operator for $A^{(k)}$ i.e.

$$S^{(k)} = I - (Q^{(k)})^{-1}A^{(k)} \tag{2.5}$$

where $Q^{(k)}$ is the lower triangular part of $A^{(k)}$ including the diagonal.

This type of algorithms has been used recently by [12, 14] within the fixed point method for image denoising problems and was found to perform satisfactorily. However, in this short paper, we shall propose a modification to the method in order to accelerate it much further.

3 An accelerated algebraic multigrid algorithm.

The improved convergence properties of the AMG over GMG comes at the cost of an AMG setup phase. In this section we outline an algorithm to try and reduce the number of setups required over the fixed point (FP) method.

3.1 Motivation.

If AMG is used as the linear solver in the fixed point method as in [12], on each step, a setup phase is required, however only a relatively small reduction in the linear residual is required, which once the setup phase has been performed will require typically 1 or 2 V-cycles. That is to say, the expensive setup phase 1 is not fully explored by phase 2. Since accurate solution of the linear equation at each fixed point step is not necessary we propose that it may be possible to recycle the AMG setup data from an earlier fixed point step for use at later fixed point steps thus reducing the computational cost of the method.

By recycling of setup data we mean the following: given a system $A\mathbf{w} = \mathbf{z}$, to solve and store interpolation operators $\hat{I}_{(2)}^{(1)}, \ldots, \hat{I}_{(L)}^{(L-1)}$ generated from the entries of a matrix $\hat{A}$ which is similar to A (in our case $A = A(\mathbf{u}^k)$ and $\hat{A} = A(\mathbf{u}^l)$, $l < k$, is the matrix from a previous fixed point step) instead of generating a new C/F-splitting and interpolation operators based on the entries of A, we use the stored interpolation operators and generate the coarse grid matrices $A^{(2)}, \ldots, A^{(L)}$ using the Galerkin principle i.e. for $p = 2, \ldots, L$ we evaluate:

$$A^{(p)} = \left(\hat{I}_{(p-1)}^{(p)}\right)^T A^{(p-1)} \hat{I}_{(p-1)}^{(p)}. \tag{3.1}$$

The coarse grid matrices with the stored transfer operators are then used in the V-cycle.

3.2 Preliminary experiments and analysis.

We first present some preliminary experiments to build our final algorithm. Since we propose to carry out the AMG setup (i.e. generation of a coarse-fine splitting and interpolation and restriction operators from matrix entries) every q fixed point steps, we may call q the frequency of data recycles on the FP steps. If $q \neq 1$ then the most recent setup data is used for the linear multigrid solver at a fixed point step; otherwise we recover the standard AMG algorithm. Our tests will be on the parameter q. In test 1, we show the effect of varying q on the overall cpu. We also address the issue of optimizing q from view points of both analysis and experiments. In test 2 for a fixed q, we examine the effect of data recycling on the speed of the inner AMG solver.

Test 1. We use the fixed point method with AMG applied to the linear system (1.9) at each step, with standard algebraic coarsening with direct interpolation and point Gauss–Seidel as smoother with 2 pre and 2 post correction smoothing steps within a multigrid V-cycle. The inner tolerance for residuals is set to be 0.1. Table 3.1 shows the number of fixed point steps required to reduce the outer (nonlinear residual) by a factor of 10^{-4} and the total cpu, for various

Table 3.1: Fixed point with AMG data for various frequencies q of setup regeneration.

q	FP Steps	cpu	Setups		V-cycles		Recycles	
			Number	Total Cpu	Number	Total cpu	Number	Total cpu
1	62	13716	62	13561	72	88	*	*
2	56	6122	28	5976	64	77	28	10
3	56	4215	19	4064	66	79	37	13
4	57	3410	15	3244	75	91	42	15
5	64	3023	13	2824	93	113	51	18
10	55	1552	6	1301	138	175	49	17
15	61	1513	5	1140	225	289	55	20
25	69	1237	3	655	364	476	66	24
50	71	1439	2	453	673	884	69	27

values of q. Also shown are the total number of setups, V-cycles and recycles with corresponding cpu in seconds. The experiments are run on a 256×256 noisy image with SNR $= 3.6$. We take $\alpha_h = 35$ and $\beta_h = 10^{-4}$.

From Table 3.1, we make the following observations. Firstly as q increases the setup cost goes down but in general we require an increasing number of V-cycles because the recycled interpolation operators become less accurate, however it appears we can achieve some reduction in the number of setups for free. Indeed, even in the case of $q = 2$ and $q = 3$, we have used fewer V-cycles than in the $q = 1$ case (standard AMG). This is due to the fact that the number of outer fixed point steps has reduced slightly but the ratio of V-cycles to fixed point steps has not increased much. In the $q = 4$ case there is only a 4% rise in the number of V-cycles. For $q \geq 5$ there is approximately a linear relationship between the increase in the number of V-cycles per fixed point step (as compared to the $q = 1$ case) and q. It is of interest to investigate the optimal frequency q_{opt}. To do this, we model the approximate increase in V-cycles by $0.15q + 0.7$ and assume that on average 1.2 V-cycles are performed on each fixed point step in the original (no-recycling) method. Denoting the cost of a setup phase by C_S, the cost of a V-cycle by C_V and the cost of a recycle by C_R we have that the average cost of a fixed point step when a setup is performed every q steps is

$$\frac{C_S}{q} + \left(1 - \frac{1}{q}\right) C_R + 1.2\, C_V (0.15q + 0.7). \tag{3.2}$$

Minimimizing this quantity we get that the optimal q is

$$q_{opt} = \frac{10}{3\sqrt{2}} \sqrt{\frac{C_S - C_R}{C_V}}. \tag{3.3}$$

Complexity-based prediction for q_{opt}. Our idea is to balance the complexity of phase 1 with that of phase 2, using the above model for the increase in V-cycles due to q. Following the Appendix, it only remains to specify the constants in the complexity analysis. From observations we take $\gamma_A = 20$, $\gamma_R = 15$ and $\gamma_P = 10$. We then have an approximate cost for the V-cycle of $300n$ and an approximate cost for the setup phase of $17/64n^2 + 730n$. If we take $n = 256^2$ the cost of a V-cycle is approximately 2×10^7 and the cost of a setup is approximately 1.19×10^9, around 90 times the cost of the V-cycle. A recycle simply involves the

evaluation of matrix RAP using stored interpolation and restriction operators. The cost of the recycling will be $(4/3\gamma_A^2+1/3\gamma_R^2)n$ which is approximately $610n$, making the cost of a V-cycle + recycling around $910n$ which for $n = 256^2$ is 6.0×10^7. Taking $C_S = 1.19 \times 10^9$, $C_V = 2 \times 10^7$ and $C_R = 6.0 \times 10^7$, we get $q_{opt} \approx 18$.

In reality the dominant cost is the cost on the finest level, on this level we know $\gamma_A = 5$ and we can assume $\gamma_R = 5$ and $\gamma_P = 4$. With these parameters we get that a V-cycle costs approximately $75n$ and a setup costs approximately $17/64n^2 + 80n$, while a recycle costs around $40n$, for $n = 256^2$ this is 5.0×10^6 flops for a V-cycle, 2.8×10^6 for a recycle and 1.15×10^9 for a setup. In this case we have $q_{opt} \approx 36$.

When we actually measure the flops associated with a V-cycle and a recycle we get that a V-cycle costs around 1.2×10^7 flops and a recycle costs 6.1×10^6, which is somewhere in between the two estimates made above.

Experiments-based prediction for q_{opt}. Since we think that our implementation of the AMG setup phase can potentially be improved further we aim throughout the paper to give a guide as to the sort of reduction in the cost of the fixed point method that can be achieved based on the relative costs of a setup, a recycle and a V-cycle. If we take $C_S = \rho_1 C_V$ and $C_R = \rho_2 C_V$ where ρ_1 is the ratio of setup cost to V-cycle cost and ρ_2 is the ratio of recycle cost to V-cycle cost then (3.3) becomes

$$q_{opt} = \frac{10}{3\sqrt{2}}\sqrt{\rho_1 - \rho_2}. \tag{3.4}$$

The factor by which the average cost of a fixed point step is decreased by is

$$\frac{\frac{C_S}{q_{opt}} + \left(1 - \frac{1}{q_{opt}}\right) C_R + 1.2C_V(0.15q_{opt} + 0.7)}{C_S + 1.2C_V} = \frac{\frac{3\sqrt{2}}{5}\sqrt{\rho_1 - \rho_2} + \rho_2 + 0.84}{\rho_1 + 1.2}. \tag{3.5}$$

We can then make a prediction of the best choice of q and the speed up in the fixed point method based on ρ_1 and ρ_2. The user can measure ρ_1 and ρ_2 in cpu time by running one AMG, or base them on complexity analysis. In our case the ρ_1 as measured in cpu is approximately 180 and ρ_2 is approximately 0.3, giving $q_{opt} \approx 31$ with a reduction in cost of around 93%.

Test 2. To get a better idea of how effective the recycling of setups is, in Table 3.2, some information on the efficiency of the linear multigrid solver for the case where $q = 10$ is given. Data are given for the first and last fixed point steps at which a particular setup is used. Shown is the number of multigrid cycles required to reduce the inner (linear) residual by a factor of 10^{-4} (rather than 0.1 as we are interested in the efficiency of the inner solver rather than the speed of the overall fixed point method) and the amount by which the residual is reduced on the first step (if no recycling is used, one V-cycle is enough to reduce the residual by a tenth, which is what we ordinarily require).

From Table 3.2 we observe that on early fixed point steps the multigrid method with recycled setup data slows down, but at later fixed point steps the perfor-

Table 3.2: Efficiency of AMG recycling for $q = 10$.

FP step	number of inner multigrid steps	convergence factor on first step
1	2	5.3×10^{-3}
10	116	1.9×10^{-1}
11	15	4.1×10^{-2}
20	38	1.2×10^{-1}
21	16	7.4×10^{-2}
30	27	9.1×10^{-2}
31	13	6.5×10^{-2}
40	12	6.4×10^{-2}

mance on the first and last fixed point steps at which a particular setup data is used is more steady, the results shown are for the case $q = 10$, but similar results have been observed for other values of q. Figure 3.1 shows the matrix entries (the matrix at the finest level) corresponding to strong connections at various fixed point steps along with the entries which corresponded to strong connections 10 fixed point steps ago, for ease of presentation the image is only 8×8 but we have

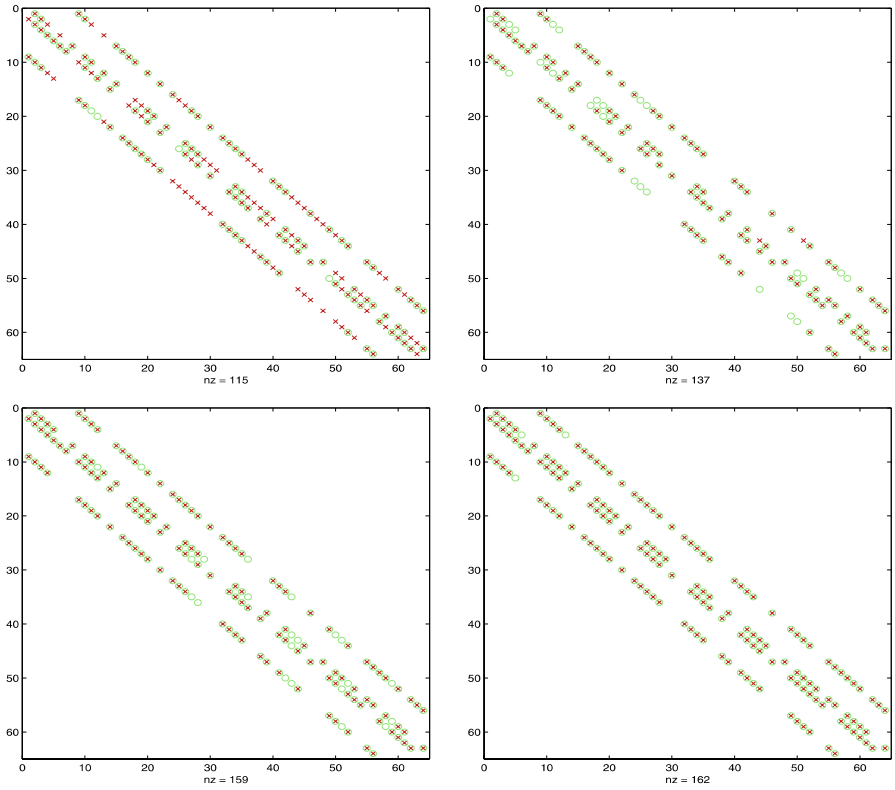

Figure 3.1: Strong connections (circles) and strong connections 10 steps ago (crosses) at steps 11, 21, 31 and 41 of the fixed point method.

observed similar results for larger images. On step 11 there are a large number of points which initially corresponded to strong connections but are now weak connections. At steps 21 and 31 there is less change in the connectivity pattern (compared to 10 steps ago) than there was at step 11 and the majority of points that have changed have gone from being weak to being strong. By step 41 there is almost no change in the pattern of strong connections compared to step 31.

The above test results suggest that more setup phases are required at early fixed point steps and less later on i.e. we should vary q in such a way that the increased work in phase 2 is gradual as the FP steps proceed.

3.3 A new method.

Indeed our new method will build on the above observations. Instead of arbitrarily carrying out a new setup phase every q FP steps, we base the decision on whether new setup data is required on the convergence history at the previous FP step. If the multigrid method takes more than ss V-cycles to reduce the linear residual by a factor of 0.1 on a particular FP step we generate new setup data at the next step. This adaptive idea allows for more setup phases on early fixed point steps when they are needed and less at later fixed point steps.

This new method should strike a balance between using as few AMG setups as possible and not causing a dramatic increase in the overall number of V-cycles required. The algorithm for the new method, which from now on we shall refer to as fixed point with AMG-R, is given below.

ALGORITHM 3.1 (Fixed Point with AMG-R).
Set $\mathbf{u}^0 = \mathbf{z}$, $ms^0 = ss + 1$, $k = 0$
While $\|\mathbf{z} - A(\mathbf{u}^k)\mathbf{u}^k\|_2 > 10^{-4}\|\mathbf{z} - A(\mathbf{u}^0)\mathbf{u}^0\|_2$
  Evaluate $A^{(1)} = A(\mathbf{u}^k)$.
  Set $\mathbf{z}^{(1)} = \mathbf{z}$, $\mathbf{v}^{(1)} = \mathbf{u}^k$, $\mathbf{r}_0^{(1)} = \mathbf{z}^{(1)} - A^{(1)}\mathbf{v}^{(1)}$.
  If $ms^k > ss$
    Perform AMG *setup and generate* $A^{(2)}, \ldots, A^{(L)}$.
    Store the Interpolation operators from the AMG *setup.*
  else
    Generate $A^{(2)}, \ldots, A^{(L)}$ *using stored*
    interpolation operators from most recent setup.
  end
  Set $ms^{k+1} = 0$
  While $\|\mathbf{r}^{(1)}\|_2 > 0.1/\|\mathbf{r}_0^{(1)}\|_2$
    $\mathbf{v}^{(1)} \leftarrow \mathrm{AMG1}^{(1)}(\mathbf{v}^{(1)}, \mathbf{z}^{(1)}, 2, 2)$
    $ms^{k+1} \leftarrow ms^{k+1} + 1$
  end
  Set $\mathbf{u}^{k+1} = \mathbf{v}^{(1)}$
  $k \leftarrow k + 1$
  end
end

Here the notation AMG1 is as defined in (2.4).

4 Numerical results.

In the following we compare the fixed point method with AMG-R, with the standard fixed point with AMG method. We give results for two 256×256 images (Lena and X-ray Fingers) each with $SNR \approx 3.5$, seen in Figure 4.1. In each case 4 different values of β_h are tested, in the case of Lena $\alpha_h = 30$ and in the case of the Fingers $\alpha_h = 35$. AMG-R in each case is run with two different (maximum) values of ss, $ss = 3$ and $ss = 10$. Shown in Tables 4.1–4.2 are the number of fixed point steps required for convergence, the total cpu time in seconds and also the total number of setups, recycles and V-cycles required, with corresponding cpu times. Clearly the new method is much faster than the standard AMG.

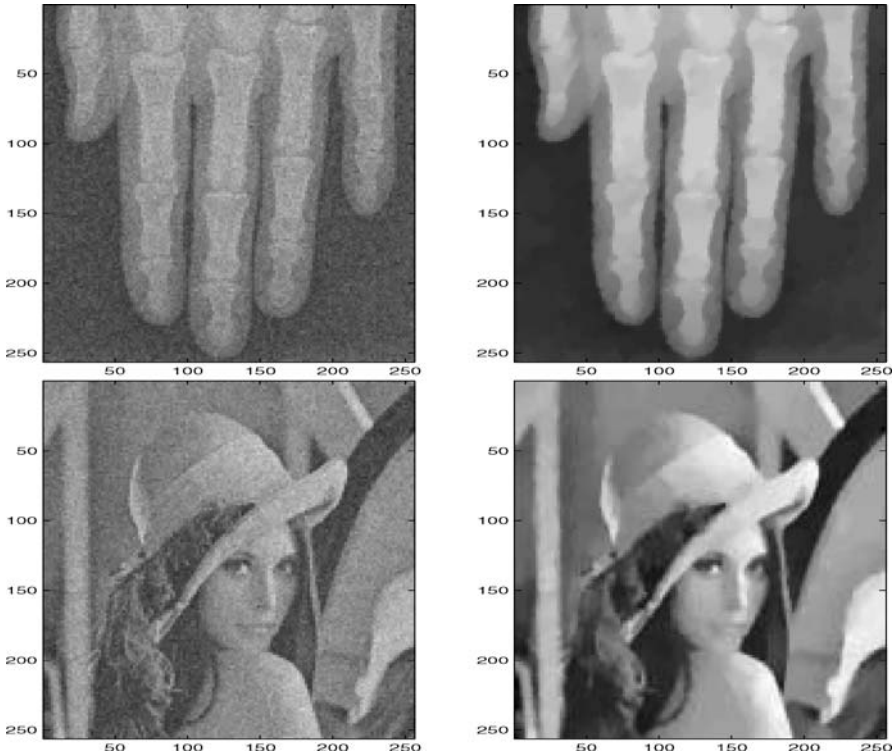

Figure 4.1: Noisy (left) and recovered (right) images using FP with the new AMG-R for 2 test examples.

The presented results in Tables 4.1–4.2 are for the cases $ss = 3$ and $ss = 10$. We give results for the case $ss = 10$ because we have found that generally this gives the best results in terms of cpu time. Given that we believe it is possible to improve upon our implementation of the AMG setup phase we also include the case $ss = 3$ as a method which should limit the increase in the number of V-cycles to be as small as possible. A value $ss = 3$ is chosen because we have observed that this is the maximum number of V-cycles required per fixed point

Table 4.1: Comparison of Fixed Point with AMG-R against Fixed Point with AMG for Lena image.

β_h	Linear Solver	AMG	AMG-R(3)	AMG-R(10)
10^{-2}	FP Setps	31	31	31
	Total cpu	7499	570	398
	Setups: No/cpu	31/7421	2/475	1/246
	V-cycles: No/cpu	31/42	40/51	80/106
	Recycles: No/cpu	*	29/11	30/13
10^{-4}	FP Setps	66	56	89
	Total cpu	14912	899	1005
	Setups: No/cpu	66/14737	3/690	2/464
	V-cycles: No/cpu	83/104	106/130	339/414
	Recycles: No/cpu	*	53/19	87/31
10^{-8}	FP Setps	172	146	148
	Total cpu	37744	2123	1937
	Setups: No/cpu	172/37165	7/1533	5/1104
	V-cycles: No/cpu	322/393	316/383	514/623
	Recycles: No/cpu	*	139/49	143/50
10^{-12}	FP Setps	310	291	272
	Total cpu	67878	3602	3340
	Setups: No/cpu	310/66747	11/2381	8/1739
	V-cycles: No/cpu	652/795	666/809	1001/1214
	Recycles: No/cpu	*	280/97	264/92

Table 4.2: Comparison of Fixed Point with AMG-R against Fixed Point with AMG for Fingers image.

β_h	Linear Solver	AMG	AMG-R(3)	AMG-R(10)
10^{-2}	FP Setps	33	31	31
	Total cpu	7979	549	396
	Setups: No/cpu	33/7895	2/454	1/239
	V-cycles: No/cpu	33/45	42/52	86/112
	Recycles: No/cpu	*	29/11	30/12
10^{-4}	FP Setps	62	56	76
	Total cpu	14094	867	846
	Setups: No/cpu	62/13923	3/662	2/449
	V-cycles: No/cpu	72/99	108/128	246/293
	Recycles: No/cpu	*	53/18	74/26
10^{-8}	FP Setps	154	173	174
	Total cpu	34520	2351	1994
	Setups: No/cpu	154/33967	8/1677	5/1061
	V-cycles: No/cpu	279/372	372/437	589/692
	Recycles: No/cpu	*	165/56	169/57
10^{-12}	FP Setps	301	310	297
	Total cpu	67427	3646	3095
	Setups: No/cpu	301/66195	11/2342	7/1485
	V-cycles: No/cpu	655/875	719/868	990/1191
	Recycles: No/cpu	*	299/103	290/100

step in the standard fixed point with AMG method. We see that in both cases, there is a significant decrease in the number of setup phases required. In the case of $ss = 3$ there is on average a 95% reduction in the number of setups compared to the fixed point with AMG method with on average only a 22% increase in the number of V-cycles. In the case of $ss = 10$ there is on average a 97% decrease in the number of setups, with an average 142% increase in the number of V-cycles. Note that in the preliminary tests where we performed a setup every q steps regardless, a 96% reduction in the number of setups ($q = 25$) led to a 406% increase in the number of V-cycles. In both cases the largest reduction in setups occurs for the smallest value of β_h, 10^{-12}, the increase in the number of V-cycles is in general also smaller for smaller β_h.

Based on these timings, the cost of the fixed point with AMG method in cpu time is reduced by around 95%. Taking the average reduction in setups and increase in V-cycles from the $ss = 3$ case and assuming that 2 V-cycles are used per fixed point step with no-recycling the expected reduction in cost for any ρ_1 and ρ_2 (as defined earlier) would be

$$\frac{0.05\rho_1 + 0.95\rho_2 + 2.44}{\rho_1 + 2}. \tag{4.1}$$

If we take $\rho_2 = 0.3$, then we can expect some speed up in the fixed point method provided the cost of a setup is a least $\rho_1 = 0.7$ times the cost of a V-cycle.

Finally we present two more experiments to test the robustness of the method. In the first test we compare fixed point with AMG against fixed point with AMG-R for a noisier version of the Lena image seen earlier (with results shown in Table 4.3); in this case the SNR=0.8 and the value of α_h is increased to 45. In the second test we run comparisons for a larger version of the Fingers Image (with results shown in Table 4.4). The size of the image is 512×512 and a similar amount of noise is present as in the 256×256 case, α_h is again chosen to be 35. In both cases we compare the methods for $\beta_h = 10^{-4}$ and $\beta_h = 10^{-8}$. In the case of the noisier Lena image we see that again both versions of the AMG-R method achieve a significant decrease in the number of setup phases, however the performance of the AMG-R(3) method is worse than in the less noisy case

Table 4.3: Comparison of AMG-R with AMG for noisier Lena image.

β_h	Linear Solver	AMG	AMG-R(3)	AMG-R(10)
10^{-4}	FP Setps	69	74	98
	Total cpu	15091	1148	1097
	Setups: No/cpu	69/14894	4/885	2/442
	V-cycles: No/cpu	97/121	134/160	401/470
	Recycles: No/cpu	*	70/25	96/34
10^{-8}	FP Setps	199	187	167
	Total cpu	42813	4210	2003
	Setups: No/cpu	199/42098	16/3395	5/1072
	V-cycles: No/cpu	399/495	439/544	567/693
	Recycles: No/cpu	*	181/65	162/56

Table 4.4: Comparison of AMG-R with AMG for the larger 512×512 Fingers image.

β_h	Linear Solver	AMG	AMG-R(3)	AMG-R(10)
10^{-4}	FP Setps	62	60	60
	Total cpu	146754	7926	6043
	Setups: No/cpu	62/146035	3/6856	2/4564
	V-cycles: No/cpu	72/407	128/698	199/1104
	Recycles: No/cpu	*	57/81	58/84
10^{-8}	FP Setps	163	175	209
	Total cpu	37997	20820	16512
	Setups: No/cpu	163/377239	8/17516	5/11429
	V-cycles: No/cpu	301/1732	406/2219	672/3757
	Recycles: No/cpu	*	167/233	204/290

particularly for the $\beta_h = 10^{-8}$ case, while the performance of the AMG-R(10) method is actually slightly better than in the less noisy case. In the case of the 512×512 fingers image, the performance (in terms of decrease in setups and increases in V-cycles) of the AMG-R methods is almost identical to the 256×256 case. The 10 fold increase in cpu time of the AMG method as compared to the 256×256 case can be potentially improved with a better implementation.

We remark that, although $ss = 10$ is a recommended (nearly optimal) parameter, its optimisation remains to be done. Equally the accuracy of the inner solver may be optimized [20] along with ss in order to improve on AMG further. In other experiments, we have found that the new AMG-R may be competitive to preconditioned conjugate gradients (PCG) methods with the incomplete Cholesky preconditioner and even the primal-dual method [10] in some cases (especially with small β). This suggests that there exist optimal hybrid methods using a mix of these methods with our AMG-R idea to help achieve robustness.

5 Conclusions.

The fixed point method with a linear GMG solver for solving the total variation denoising PDE is not robust with respect to the smoothing parameter β. The usually robust AMG solver comes with a high computational cost. We have proposed a method for accelerating the fixed point method, when AMG is used as the inner linear solver, by recycling the AMG setup data adaptively: an AMG setup phase was performed only when the number of V-cycles required on the previous fixed point step exceeded some user-defined parameter ss. Experiments have shown that a significant decrease in the total number of setups can be achieved, for only a modest increase in the number of V-cycles performed, for a wide range of values of the parameter β_h. With our current implementation we have reduced the overall cost of the method by up to 20 times. Much further refinement is possible but even with a very efficient implementation of the AMG setup e.g. $\rho_1 = 4$ we would expect a further halving of the costs with our AMG-R approach.

Appendix – Complexity estimation for the AMG Algorithm 2.1.

In this Appendix we present an analysis of the costs associated with implementing the Algorithm 2.1 with the standard *C/F* splitting algorithm and direct interpolation. As an exact analysis is not possible due to the nature of AMG, we shall make some reasonable assumptions. We will use the following notation in the cost analysis that follows

$n^{(k)} =$ number of points on level k

$n^{F^k} =$ number of fine points on level k

$n^{C^k} = n^{(k+1)} =$ number of coarse points on level k

$\gamma_{A^k} =$ max number of entries in a row of A^k

$\gamma_{R^k} =$ max number of entries in a row of $R^k = I_k^{k+1}$ (restriction $k \to k+1$)

$\gamma_{P^k} =$ max number of entries in a row of $P^k = I_{k+1}^k$ (prolongation $k+1 \to k$).

Also finding all the nonzero entries in a set or finding the maximum value in set is modeled by flops equal to twice the size of the set.

A.1 The AMG *phase* 1.

A.1.1 Cost of finding neighbours and strong connections.

To find N_i we must search row i of the matrix for non-zero, non-diagonal entries. In reality the position of all non-zero entries in a sparse matrix can be found very cheaply in MATLAB so we ignore this cost here. Once we have the set of neighbours we have to find the maximum value of $-a_{ij}$ over $j \in N_i$ and then find S_i. We have to search through the set $\{-a_{ij} | j \in N_i\}$ to find the maximum, and then search the set again to find the entries greater than θ times the maximum, since an upper bound on the size of N_i is $\gamma_{A^k} - 1$ an upper bound for this cost is $4(\gamma_{A^k} - 1)$. An upper bound on the cost of finding all N_i and S_i on level k is therefore

$$n^{(k)} \left[4(\gamma_{A^k} - 1)\right]. \tag{5.1}$$

A.1.2 The C/F splitting algorithm.

We assume that the size of $C^k = n^{(k+1)} = 1/4n^{(k)}$ i.e. assume standard coarsening or $n^{(k)} = n/4^{k-1}$ for $1 \le k \le L$. We therefore go around the C/F splitting loop $1/4n$ times and on average each coarse point defines 3 fine points i.e. the size of $S_i^T \cap U$ is 3. Although in our experience based on cpu analysis similar to above a reasonable assumption is that the cost of finding $S \cap U$ where S is a small set, is around 3 times the size of U we can in our current implementation find $S_i^T \cap U$ simply by finding the nonzero entries of a set the size of S_i^T, which we assume below has size 4. Based on these assumptions we have the following

estimate for the cost of performing the C/F split on level k:

$$\sum_{m=0}^{1/4n^{(k)}-1} [2(n^{(k)} - 4m) + 8] + 3\,(8+4) + 8 + 4. \tag{5.2}$$

The first term comes from finding $i \in U$ with maximal λ_i and then finding $j \in S_i^T \cap U$, the second term comes from finding $l \in S_j \cap U$ setting $\lambda_l = \lambda_l + 1$ for each $j \in S_i^T \cap U$ (assume that the size of S_j and $S_j \cap U$ is 4) the third term comes from finding $j \in S_i \cap U$ and the last term setting $\lambda_j = \lambda_j - 1$ (again assume S_i and $S_i \cap U$ have 4 members). An approximate cost estimate is therefore

(5.3)

$$(1/4n)(2n + 56) - 8 \sum_{m=0}^{1/4n^{(k)}-1} m \approx 1/2n^2 + 14n - 8(1/8n)(1/4n) = 1/4n^2 + 14n.$$

A.1.3 The direct interpolation.

On level k direct interpolation involves finding for each $i \in F^k$, $\alpha_i = \frac{\sum_{j\in N_i} a_{ij}^-}{\sum_{k\in P_i} a_{ik}^-}$ and $\beta_i = \frac{\sum_{j\in N_i} a_{ij}^+}{\sum_{k\in P_i} a_{ik}^+}$ or if P_i^+ is empty finding α_i and setting $a_{ii} = a_{ii} + \sum_{j\in N_i} a_{ij}^+$. Then defining the interpolation weights as

$$w_{ik} = \begin{cases} -\alpha_i a_{ik}/a_{ii} & k \in P_i^- \\ -\beta_i a_{ik}/a_{ii} & k \in P_i^+ \end{cases} \tag{5.4}$$

and denoting by n^{P^k} the max size of P_i over all $i \in F^k$ and by n^{N^k} the max size of N_i over all $i \in F^k$, an upper bound for the cost of the direct interpolation on level k is

$$n^{F^k}[n^{N^k} + 3n^{P^k}]. \tag{5.5}$$

A.1.4 Cost of the Galerkin matrix RAP.

Given that we expect that A and P will be very sparse most row_A, $column_P$ multiplications will produce zeros. A cost of $\gamma_{A^k}^2 n^{(k)}$ is a reasonable assumption for the cost of evaluating $A^k P^k$ similarly $\gamma_{R^k}^2 n^{C^k}$ is a reasonable assumption for the cost of multiplying $A^k P^k$ by R^k.

A.1.5 Overall complexity of phase 1.

Putting all of this together, noting $S_{N^k} = \gamma_{A^k} - 1$ and assuming $S_{P^k} = 4$, we have as an estimate for the cost of an AMG setup phase the following:

(5.6)

$$\begin{aligned}
&\sum_k 4(\gamma_{A^k}-1)n^{(k)} + 1/4(n^{(k)})^2 + 14n^{(k)} + n^{F^k}(\gamma_A+11) + \gamma_A^2 n^{(k)} + \gamma_R^2 n^{C^k} \\
&\approx \sum_{l=0}^{\infty} 1/4(1/16)^l n^2 + 3/4(1/4)^l\big(17/3\gamma_A + 4/3\gamma_A^2 + 1/3\gamma_R^2 + 8\big)n \\
&= 17/64n^2 + \big(17\gamma_A + 4\gamma_A^2 + \gamma_R^2 + 24\big)n/3,
\end{aligned}$$

where we take γ_A, γ_R, γ_P respectively as the maximum value over all levels of γ_{A^k}, γ_{R^k}, γ_{P^k}.

A.2 The AMG *phase* 2.

Once the setup phase has been performed the costs associated with the V-cycle on level k are: **(1)** the cost of 4 Gauss–Seidel steps each of which involves solving $Mv^{new} = (Nv^{old} + f)$ where M is the lower triangular part of A^k including the diagonal and N is the upper triangular part of A. Assuming that γ_A^k is small and that the entries of A are split evenly about the diagonal and using the fact that the cost of inverting or multiplying by a $n \times n$ triangular matrix with a small number, say σ, of entries on each row is approximately $(2\sigma - 1)n$ we have $4\left[2(\gamma_A^k - 1)n + n\right]$ as an estimate for the cost of the Gauss–Seidel steps; **(2)** the cost of the residual calculation an upper bound for which is $n^{(k)} + 2\gamma_{A^k} n^{(k)}$; **(3)** the cost of restricting the residual, an upper bound for which is $2\gamma_{R^k} n^{C^k}$; **(4)** the cost of interpolating the error back from the coarse grid and correcting, an upper bound for which is $2\gamma_{P^k} n^{F^k} + n^{C^k} + n^{(k)}$. Finally we have as an upper bound for the cost of a V-cycle:

$$\sum_{\ell=0}^{\infty}(1/4)^{\ell}[10\gamma_A + 1/2\gamma_R + 3/2\gamma_P - 7/4]n = (40\gamma_A + 2\gamma_R + 6\gamma_P - 7)n/3. \tag{5.7}$$

Acknowledgements.

The authors thank all three anonymous referees for making helpful remarks and suggestions.

REFERENCES

1. W. Briggs, *A Multigrid Tutorial*, SIAM, Philadelphia, USA, 1987.
2. P. Blomgren, T. F. Chan, P. Mulet, L. Vese, and W. L. Wan, *Variational PDE models and methods for image processing*, in Res. Notes Math., vol. 420, pp. 43–67, Chapman & Hall/CRC, (2000).
3. R. H. Chan, T. F. Chan, and W. L. Wan, *Multigrid for differential convolution problems arising from image processing*, in Proc. Sci. Comput. Workshop, eds. R. Chan, T. F. Chan and G. H. Golub, Springer, (1997). See also CAM report 97-20, UCLA, USA.
4. R. H. Chan, T. F. Chan, and C. K. Wong, *Cosine transform based preconditioners for total variation minimization problems in image processing*, IEEE Trans. Image Proc., 8 (1999), pp. 1472–1478. See also CAM report 97–44, UCLA.

5. R. H. Chan, Q. S. Chang, and H. W. Sun, *Multigrid method for ill-conditioned symmetric Toeplitz systems*, SIAM J. Sci. Comput., 19 (1998), pp. 516–529.
6. R. H. Chan, T. F. Chan, and W. L. Wan, *Multigrid for differential-convolution problems arising from image processing*, Proceedings of the Workshop on Scientific Computing, vol. 97, Springer, (1997).
7. T. F. Chan and K. Chen, *On a nonlinear multigrid algorithm with primal relaxation for the image total variation minimisation*, Numer. Algorithms., 41 (2006), pp. 387–411.
8. T. F. Chan and K. Chen, *An optimization based total variation image denoising*, SIAM J. Multiscale Modeling & Simulation, 5(2) (2006), pp. 615–645.
9. T. F. Chan, K. Chen, and X.-C. Tai, *Nonlinear multilevel schemes for solving the total variation image minimization problem*, in Image Processing Based On Partial Differential Equations, eds. X.-C. Tai, K.-A. Lie, T.F. Chan, S. Osher, pp.265–288, Springer, (2006).
10. T. F. Chan, G. H. Golub, and P. Mulet, *A nonlinear primal-dual method for total variation-based image restoration*, SIAM J. Sci. Comput., 20 (1999), pp. 1964–1977.
11. T. F. Chan, H. M. Zhou, and R. H. Chan, *Continuation method for total variation denoising problems*, UCLA CAM Report, USA, (1995).
12. Q. S. Chang and I. Chern, *Acceleration methods for total variation based image denoising*, SIAM J. Sci. Comput., 25 (2003), pp. 982–994.
13. Q. S. Chang, Y. S. Wong, and H. Fu, *On the algebraic multigrid method*, J. Comput. Phys., 125 (1996), pp. 279–292.
14. Q. S. Chang, W. C. Wang, and J. Wu, *A method for total variation-based reconstruction of noisy and blurred images*, in: Image Processing Based On Partial Differential Equations, eds. X.-C. Tai, K.-A. Lie, T.F. Chan, and S. Osher, pp. 95–108, Springer, (2006).
15. K. Chen, *Matrix Preconditioning Techniques and Applications*, Cambridge Monographs on Applied and Computational Mathematics (No. 19). Cambridge University Press, UK, (2005).
16. C. Frohn-Schauf, S. Henn, and K. Witsch, *Nonlinear multigrid methods for total variation image denoising*, Comput. Visual Sci., 7 (2004), pp. 199–206.
17. M. Hintermüller and K. Kunisch, *Total bounded variation regularization as a bilaterally constrained optimization problem*, SIAM J. Appl. Math., 64 (2004), pp. 1311–1333.
18. W. Hinterberger, M. Hintermüller, K. Kunisch, M. von Oehsen, and O. Scherzer, *Tube methods for BV regularization*, J. Math. Imaging Vis., 19 (2003), pp. 219–235.
19. T. Kärkkäinen, K. Majava, and M. M. Mäkelä, *Comparison of formulations and solution methods for image restoration problems*, Series B Report No. B 14/2000, Department of Mathematical Information Technology, University of Jyväskylä, Finland, (2000).
20. I. E. Karpin and O. Axelsson, *On a class of nonlinear equation solvers based on residual norm reduction over a sequence of affine subspaces*, SIAM J. Numer. Anal., 16 (1995), pp. 228–249.
21. Y. Y. Li and F. Santosa, *A computational algorithm for minimizing total variation in image restoration*, IEEE Trans. Image Proc., 5 (1996), pp. 987–995.
22. A. Marquina and S. Osher, *Explicit algorithms for a new time dependant model based onlevel set motion for nonlinear deblurring and noise removal*, SIAM J. Sci. Comput., 22 (2000), pp. 387–405.
23. L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D, 60 (1992), pp. 259–268.
24. J. W. Ruge and K. Stuben, *Algebraic Multigrid*, in S. F. McCormick (ed.), Multigrid Methods, SIAM, Philadelphia, USA (1987).
25. J. Savage and K. Chen, *An improved and accelerated nonlinear multigrid method for total-variation denoising*, Int. J. Comput. Math., 82 (2005), pp. 1001–1015.
26. J. Savage and K. Chen, *On multigrids for solving a class of improved total variation based PDE models*, in Image Processing Based On Partial Differential Equations, X.-C. Tai, K.-A. Lie, T.F. Chan, and S. Osher, eds., pp. 69–94, Springer (2006).

27. G. Steidl, J. Weickert, T. Brox, P. Mrázek, and M. Welk, *On the equivalence of soft wavelet shrinkage, total variation diffusion, total variation regularization, and SIDEs*, SIAM J. Numer. Anal., 42(2) (2004), pp. 686–713.

28. U. Trottenberg, C. Oosterlee, and A. Schuller, *Multigrid*, Academic Press, London (2001). (See its Appendix on AMG by K. Stuben).

29. C. R. Vogel, *A multigrid method for total variation-based image denoising*, In K. Bowers and J. Lund, eds., *Computation and Control IV*, vol. 20, Progress in Systems and Control Theory, Birkhäuser (1995).

30. C. R. Vogel, *Negative results for multilevel preconditioners in image deblurring*, in M. Nielson et al., eds., Scale-space theories in computer vision, pp. 292–304, Springer (1999).

31. C. R. Vogel and M. E. Oman, *Iterative methods for total variation denoising*, SIAM J. Sci. Stat. Comput., 17 (1996), pp. 227–238.

32. C. R. Vogel and M. E. Oman, *Fast, robust total variation-based reconstruction of noisy, blurred images*, IEEE Trans. Image Proc., 7 (1998), pp. 813–824.

33. C. R. Vogel, *Computational Methods for Inverse Problems*, SIAM, Philadelphia, USA, 2002.

34. C. Wagner, *Introduction to Algebraic Multigrid*, Course Notes of an Algebraic Multigrid, Course at the University of Heidelberg in the Winter semester, 1998/1999.

35. T. Washio and C. Oosterlee, *Krylov subspace acceleration for nonlinear multigrid schemes*, Electronic Trans. Numer. Anal., 6 (1997), pp. 271–290.

36. P. Wesseling, *An Introduction to Multigrid Methods*, Wiley, Chichester, UK, 1992.