Multigrid and Multilevel Preconditioners for Computational Photography

Dilip Krishnan
Department of Computer Science
New York University
dilip@cs.nyu.edu

Richard Szeliski
Interactive Visual Media Group
Microsoft Research
szeliski@microsoft.com

New York University
Technical Report NYU-TR-941
October, 2011

Abstract

This paper unifies multigrid and multilevel (hierarchical) preconditioners, two widely-used approaches for solving computational photography and other computer graphics simulation problems. It provides detailed experimental comparisons of these techniques and their variants, including an analysis of relative computational costs and how these impact practical algorithm performance. We derive both theoretical convergence rates based on the condition numbers of the systems and their preconditioners, and empirical convergence rates drawn from real-world problems. We also develop new techniques for sparsifying higher connectivity problems, and compare our techniques to existing and newly developed variants such as algebraic and combinatorial multigrid. Our experimental results demonstrate that, except for highly irregular problems, adaptive hierarchical basis function preconditioners generally outperform alternative multigrid techniques, especially when computational complexity is taken into account.

Contents

1 Introduction

Multigrid and multilevel preconditioning techniques have long been widely used in computer graphics and computational photography as a means of accelerating the solution of large gridded optimization problems such as geometric modeling (Gortler and Cohen 1995), high-dynamic range tone mapping (Fattal, Lischinski, and Werman 2002), Poisson and gradient-domain blending (Pérez, Gangnet, and Blake 2003; Levin, Zomet, Peleg *et al.* 2004; Agarwala *et al.* 2004), colorization (Levin, Lischinski, and Weiss 2004) (Fig. 11), and natural image matting (Levin, Lischinski, and Weiss 2008). They have also found widespread application in the solution of computer vision problems such as surface interpolation, stereo matching, optical flow, and shape from shading (Terzopoulos 1983, 1986; Szeliski 1990, 1991; Pentland 1994; Yaou and Chang 1994; Lai and Vemuri 1997), as well as large-scale finite element and finite difference modeling (Briggs, Henson, and McCormick 2000; Trottenberg, Oosterlee, and Schuller 2000).

While the locally adaptive hierarchical basis function technique developed by Szeliski (2006) showed impressive speedups over earlier non-adaptive basis functions (Szeliski 1990), it was never adequately compared to state-of-the art multigrid techniques such as algebraic multigrid (Briggs, Henson, and McCormick 2000; Trottenberg, Oosterlee, and Schuller 2000) or to newer techniques such as lattice-preserving multigrid (Grady 2008) and combinatorial multigrid (Koutis, Miller, and Tolliver 2009). Furthermore, the original technique was restricted to problems defined on four neighbor ($\mathcal{N}_4$) grids. On the other hand, it was shown that the adaptive hierarchical basis function technique was superior to traditional preconditioners such as MILU, ILU0 and RILU (Saad 2003), and therefore we do not consider those preconditioners in this paper.

In this paper, we generalize the sparsification method introduced in (Szeliski 2006) to handle a larger class of grid topologies, and show how multi-level preconditioners can be enhanced with smoothing to create hybrid algorithms that accrue the advantages of both adaptive basis preconditioning and multigrid relaxation. We also provide a detailed study of the convergence properties of all of these algorithms using both condition number analysis and empirical observations of convergence rates on real-world problems in computer graphics and computational photography.

The scope of this paper is limited to quadratic energy minimization problems defined on regular gridded lattices such as those used in computer graphics, computational photography, and physical simulations.[1] On such lattices, we support both nearest neighbor ($\mathcal{N}_4$) and eight neighbor ($\mathcal{N}_8$) connectivity. The quadratic energy being minimized (Eqn. 2) has a *Hessian* or *quadratic form* matrix A where the off-diagonal elements are potentially non-zero (and negative) only for $\mathcal{N}_4$ or $\mathcal{N}_8$ neighbors on the grid and the diagonals are always positive and greater than or equal to the sum of the negative off-diagonals. Section 2 explains how

[1] When viewed as priors for Bayesian inferences, such problems correspond to Gaussian Markov Random Fields (GMRFs).

such problems arise in a variety of computer graphics and computational photography applications.

Matrices where $a_{ii} \geq \sum_{j,j\neq i} |a_{ij}|$ are known as *symmetric diagonally dominant* (SDD) matrices. Although algebraic and combinatorial multigrid techniques can solve SDD system on more general graphs and triangulations (and also with positive off-diagonal elements (Koutis, Miller, and Tolliver 2009)), we do not address such problems in this paper.

Our experimental results demonstrate that locally adaptive hierarchical basis functions (ABF) (Szeliski 2006) combined with the extensions proposed in this paper generally outperform algebraic and combinatorial multigrid techniques, especially once computational complexity is taken into account. However, for highly irregular and inhomogeneous problems, techniques that use adaptive coarsening strategies (Section 3), which our approach does not currently use, may perform better.

In this paper, we consider general spatially varying quadratic cost functions. This allows the algorithms presented here to be applied to a variety of problems. In the existing literature, a number of optimized schemes have been developed focusing on specific problems. We give a brief survey of such schemes here.

Agarwala (2007) considers the problem of Poisson blending when applied to the seamless stitching of very large images. The optimization proceeds by defining a variable-sized grid (large spacing in smooth regions and small spacing in non-smooth regions), resulting in a much smaller linear system. McCann and Pollard (2008) use a standard V-cycle geometric multigrid method on a GPU, to achieve near real-time gradient field integration. Farbman, Hoffer, Lipman *et al.* (2009) solve the image cloning problem by replacing the linear system solver with an adaptive interpolation and mesh refinement strategy. This is similar in spirit to the work of Agarwala (2007). However, this approach cannot work for problems such as HDR image compression or Poisson blending with mixed gradients.

The algorithm in (Roberts 2001) is similar to that of (Szeliski 1990) but with some important differences in the preconditioner construction (there is no diagonal preconditioning of fine-level variables and Jacobi smoothing is used). This preconditioner-based solver shows better performance than geometric multigrid and extensions to 3D problems are given. However, since the interpolants aren't adapted to the problem, for inhomogeneous problems, both (Roberts 2001) and (Szeliski 1990) will underperform the adaptive basis functions presented in (Szeliski 2006) and this paper. Wang, Raskar, and Ahuja (2004) use the solver of Roberts (2001) to solve a 3D version of the Poisson blending problem for video. Jeschke, Cline, and Wonka (2009) present a GPU solver for the integration of a homogeneous Poisson equation; this method uses Jacobi iterations with modified stencil sizes. McAdams, Sifakis, and Teran (2010) present a parallelized Poisson solver for fluid simulation problems. This solver is based on geometric multigrid. The algorithms in (Wang, Raskar, and Ahuja 2004; Jeschke, Cline, and Wonka 2009; McAdams, Sifakis, and Teran 2010) are all

restricted to homogeneous Poisson problems and their extension to inhomogeneous problems does not seem straightforward. Kazhdan, Surendran, and Hoppe (2010) develop a high-performance solver where their key technical contribution is to parallelize the raster-order Gauss-Seidel smoothing. Our ABF preconditioner and four-color Gauss-Seidel smoothers do not require such streaming implementations, as simpler overlapped (multi-resolution) tiles can be used to perform out-of-core computations.

The remainder of this paper is structured as follows. Section 2 presents the general class of regular gridded problems that we study, along with the associated quadratic energy we minimize and the linear systems of equations we aim to solve. In Section 3, we describe the various solvers we evaluate and qualitatively compare their characteristics. Section 4 describes how the the eigenvalues of the iterative solvers and condition numbers of the preconditioned solvers can be used to predict the convergence rates of various algorithms. In Section 5, we show how to extend the sparsification step introduced in (Szeliski 2006) to a wider range of problems and how condition number analysis can be used to derive such sparsifiers. In Section 6, we give descriptions of the sample problems that we study. Section 7 contains our detailed experimental analysis of both the theoretical (condition number) and empirical convergence rates of the various algorithms and variants we consider. Section 8 contains conclusions and recommendations. To allow the community to better understand and use these solvers, we provide a MATLAB implementation of these algorithms at `www.cs.nyu.edu/~dilip/research/abf`

2 Problem formulation

Following (Szeliski 2006), we consider two-dimensional variational problems discretized on a regular grid. We seek to reconstruct a function f on a discrete domain Ω given data d and, optionally, gradient terms g^x and g^y. Let $(i, j) \in \Omega$ represent a point in this domain. The problems we study involve finding the solution f that minimizes the quadratic energy

$$E_5 = \sum_{i,j \in \Omega} w_{i,j}(f_{i,j} - d_{i,j})^2 + s^x_{i,j}(f_{i+1,j} - f_{i,j} - g^x_{i,j})^2 + s^y_{i,j}(f_{i,j+1} - f_{i,j} - g^y_{i,j})^2. \quad (1)$$

The $w_{i,j}$ are (non-negative, potentially spatially-varying) data term weights and the $s^x_{i,j}$ and $s^y_{i,j}$ are (non-negative, potentially spatially-varying) gradient term weights corresponding to the x and y directions respectively. These weights are used to balance the closeness of the values of f to d and the values of the gradients of f to g^x and g^y. The target gradient values g^x and g^y can be set to 0 if only smoothing is desired. (See Section 6 on how these weights map to various computer graphics and computational photography problems.)

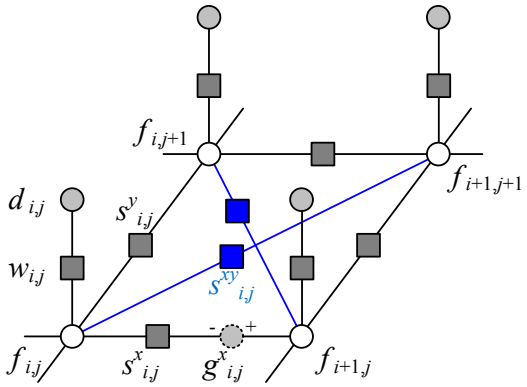


Figure 1: Resistive grid interpretation of energy E_5 and E_9. The weights $w_{i,j}$ and smoothness terms $s_{i,j}$ correspond to conductances (inverse resistances), while the data terms $d_{i,j}$ and optional $g_{i,j}$ correspond to bias voltages. The two blue boxes show the additional terms present in E_9.

If we represent the values in the function f by a vector x, we can then re-write Eqn. 1 as a quadratic energy,

$$E = x^T Ax - 2b^T x + c, \tag{2}$$

where A is a sparse symmetric positive definite (SPD) and symmetric diagonally dominant (SDD) matrix with only non-positive off-diagonal terms. Taking the derivative of this energy with respect to x and setting it to 0 gives us the linear systems of equations to be solved,

$$Ax = b. \tag{3}$$

The horizontal and vertical finite differences in Eqn. 1 give A a 5-banded structure.

If we add xy cross-derivatives, A has a 9-banded structure. The energy function becomes

$$E_9 = E_5 + \sum_{i,j \in \Omega} s^{xy}_{i,j}(f_{i+1,j-1} - f_{i,j} - g^{xy}_{i,j})^2 + s^{yx}_{i,j}(f_{i+1,j+1} - f_{i,j} - g^{yx}_{i,j})^2. \tag{4}$$

In this paper, we consider the solution of linear systems involving both 5-banded and 9-banded matrices. We often refer to these systems as 5-point stencils and 9-point stencils, respectively.

It is easy to show that these quadratic energies are equivalent to a resistive grid with bias voltages $d_{i,j}$ being optionally present at each node and bias voltages $g_{i,j}$ being optionally present along some edges in the grid, as shown in Fig. 1. The weights $w_{i,j}$ and smoothness terms $s_{i,j}$ correspond to conductances (inverse resistances). This resistive grid analogy has been used to re-interpret the system as a diffusion or random walk process (Grady 2006) and also to compute condition number estimates using Rayleigh quotients (Section 4)(Koutis, Miller, and Tolliver 2009).

3 Solution techniques

In this section, we provide details of the methods used to solve systems of equations of the form $Ax = b$ when A is a sparse, banded, symmetric positive definite matrix. While for some techniques such as direct solvers, we only provide brief descriptions, for iterative solvers, we go into more depth to describe the algorithms, since they are of interest to us here.

Traditional dense direct techniques such as Cholesky decomposition and Gaussian elimination (Golub and Van Loan 1996) have a cost of $O(n^3)$, where n is the number of variables. For gridded two-dimensional problems such as the ones we study in this paper, sparse direct methods such as nested dissection (Davis 2006; Alon and Yuster 2010) cost $O(n^{3/2})$. When n becomes large, such direct solvers incur too much memory and computational cost to be practical.

In many graphics and vision applications, it is not necessary to solve the linear systems exactly. Approximate solutions with a medium level of accuracy are often sufficient, owing to the eventual discretization of the output value f (e.g., a color image) into integral values.[2] Hence, we can use iterative methods, which have the advantage of allowing termination when a pre-specified level of accuracy has been reached (Saad 2003).

Simple iterative methods such as Jacobi and Gauss-Seidel have $O(n)$ cost per iteration. The number of iterations depends on the condition number κ of A and on the desired level of accuracy (Section 4). For the kinds of reconstruction and interpolation problems studied in this paper, the condition numbers can be very large, requiring far too many iterations to be practical. However, iterative methods such as Jacobi and Gauss-Seidel, when used as *smoothers*, are an inexpensive way to discover high-frequency components of the correction in very few iterations. Hence they are used in multigrid methods in conjunction with a pyramid of grids.

Conjugate Gradient (CG) algorithms typically exhibit much faster convergence than Jacobi or Gauss-Seidel methods for SPD problems (Shewchuk 1994; Saad 2003). CG is almost always used with a preconditioner, and preconditioned CG (PCG) usually requires many fewer iterations than Jacobi to reach the same accuracy. The main challenge with PCG is the design of preconditioners that are computationally efficient and yet achieve a significant acceleration over unpreconditioned CG.

Multigrid (MG) methods (Briggs, Henson, and McCormick 2000; Trottenberg, Oosterlee, and Schuller 2000) are an alternative family of numerically stable and computationally efficient methods for iteratively solving SPD linear systems. Originally developed for homogeneous elliptic differential equations, MG

[2] Even when the output is used for physical simulation or animation, accuracy requirements are often bounded and moderate.

methods now constitute a family of methods under a common framework that can be used to solve both inhomogeneous elliptic and non-elliptic differential equations. This family includes algebraic multigrid (AMG) (Briggs, Henson, and McCormick 2000; Trottenberg, Oosterlee, and Schuller 2000; Kushnir, Galun, and Brandt 2010; Napov and Notay 2011) and combinatorial multigrid (CMG) techniques (Koutis, Miller, and Tolliver 2009).

In this section, we review classic single-level relaxation algorithms such as Jacobi and Gauss-Seidel (Section 3.1) and conjugate gradient descent (Section 3.2). We then describe multilevel preconditioners (Section 3.3). Next, we describe geometric multigrid techniques (Section 3.4), algebraic multigrid (Section 3.5), and combinatorial multigrid (Section 3.6). Experimental comparisons of all these approaches are presented in Section 7.

In this paper, we consider fixed coarsening approaches of two kinds: full-octave and half-octave. Consider a fine-level grid with N points. In full-octave coarsening of a two-dimensional grid, every fourth point is assigned to the coarse grid. The coarse level points are evenly spread across the grid. In half-octave coarsening, the fine-level grid is divided into two disjoint sets of interleaved points, called red and black points. There are equal number of red and black points ($N/2$). The red points are then assigned to the coarse grid. Coarsening (whether full-octave or half-octave) is then performed recursively to give a full multilevel or multigrid pyramid. A multilevel pyramid constructed using red-black coarsening is depicted in Fig. 2.

3.1 Jacobi and Gauss-Seidel smoothing

Smoothing (or *relaxation*) techniques, such as Jacobi, Gauss-Seidel, and *Richardson iteration* (Saad 2003), apply a series of fixed update steps to the current state vector x based on the value of the current residual

$$r = b - Ax, \tag{5}$$

which is also the direction of steepest descent.

As described in more detail in Appendix A and written programmatically in Algorithm 1, ω-damped Jacobi smoothing involves dividing the residual r by the diagonal D of the A matrix and then taking a step of size ω in this direction.[3] Jacobi smoothing is completely parallel, since at each iteration, every point may be updated independently of all others.

There are several variants of Gauss-Seidel smoothing, which involves updating some variables while keeping certain others fixed. When used in isolation, raster-order Gauss-Seidel does a better job of propa-

[3] A value of $\omega < 1$ is usually required to damp the highest-frequency oscillations. For uniformly weighted (homogeneous isotropic) problems, a value of $\omega = 0.8$ is a good choice (Trottenberg, Oosterlee, and Schuller 2000).

Algorithm 1 Jacobi smoothing

$[x_\nu] = \textit{Smooth}(x_0, A, b, \omega, \nu)$

INPUT: Initial solution x_0, matrix A, r.h.s. b, damping factor ω, number of iterations ν

OUTPUT: Smoothed solution x_ν

1. **for** $k = 1 \ldots \nu$

2. $\quad r_{k-1} = b - Ax_{k-1}$

3. $\quad x_k = x_{k-1} + \omega D^{-1} r_{k-1}$ // D is diagonal of A

4. **end for**

gating long-range interactions than Jacobi iteration. When used as part of multigrid, red-black Gauss-Seidel (GS-RB), where the red nodes in a checkerboard are updated first, followed by the black ones (Saad 2003) does a better job of attenuating high frequencies and is also more amenable to fine-grained data-parallel implementations since the red nodes may all be updated in parallel. However, GS-RB can only be used with 5-point stencils, which are not preserved with most multigrid coarsening schemes. In the case of 9-point stencils, a four-color variant of Gauss-Seidel (GS-FC) can be used, and this is the preferred technique we recommend (Trottenberg, Oosterlee, and Schuller 2000, p 173). Algorithm 1 and Algorithm 2 provide the algorithms for Jacobi and GS-FC smoothing, respectively.

Algorithm 2 Gauss-Seidel four color smoothing

$[x_\nu] = \textit{Smooth}(x_0, A, b, \omega, \nu)$

INPUT: Initial solution x_0, matrix A, r.h.s. b, damping factor ω, number of iterations ν

OUTPUT: Smoothed solution x_ν

1. Divide points in x_0 into 4 disjoint sets: F_1, F_2, F_3, F_4. Let $F = \cup F_i$

2. **for** $k = 1 \ldots \nu$

3. $\quad x_k = x_{k-1}$

4. $\quad$ **for** $i = 1 \ldots 4$

5. $\quad\quad$ Set $\tilde{A} = A(F_i, F - F_i)$ and $\tilde{x} = x_k(F - F_i)$

6. $\quad\quad r_{k-1}(F_i) = b(F_i) - \tilde{A}\tilde{x}$ // Update the ith color

7. $\quad\quad x_k(F_i) = x_{k-1}(F_i) + \omega D^{-1}(F_i, F_i) r_{k-1}(F_i)$ // D is diagonal of A

8. $\quad$ **end for**

9. **end for**

Used in isolation, however, none of these smoothing techniques perform well when applied to the typ-

ical kinds of interpolation or reconstruction problems that arise in computer graphics and computational photography.[4] The reason for this is that while these smoothers are very good at quickly reducing highly oscillatory components of the error (Fig. 3a). they are much slower at reducing smooth components (Briggs 1987). As a result, these smoothers must be coupled with multigrid techniques to speed up convergence.

3.2 Preconditioned conjugate gradient descent

Preconditioned conjugate gradient descent (PCG) (Shewchuk 1994) is a popular iterative algorithm for the solution of SPD linear systems. In Algorithm 3, we provide details of PCG.

Algorithm 3 Preconditioned conjugate gradient descent

$[x] = PCG(x_0, A, b, M)$

INPUT: Initial solution x_0, matrix A, r.h.s. b, SPD preconditioner M

OUTPUT: Solution x of $Ax = b$

1. $r_0 = b - Ax_0$
2. $d_0 = Mr_0$
3. **for** $k = 1, 2, \ldots$
4. $\quad \alpha_k = (d_{k-1}^T r_{k-1})/(d_{k-1}^T A d_{k-1})$
5. $\quad x_k = x_{k-1} + \alpha_k d_{k-1}$
6. $\quad r_k = b - Ax_k = r_{k-1} - \alpha_k A d_{k-1}$
7. $\quad \tilde{r}_k = Mr_k$ // Preconditioning
8. $\quad \beta_k = (\tilde{r}_k^T A d_{k-1})/(d_{k-1}^T A d_{k-1})$
9. $\quad d_k = \tilde{r}_k - \beta_k d_{k-1}$
10. **end for**

Notice how this algorithm contains the following major components:

- computing an initial residual $r_0 = b - Ax_0$;
- preconditioning the residual using the matrix M, $\tilde{r}_k = Mr_k$;
- computing the step size α_k and the conjugation parameter β_k using a matrix multiply Ad_{k-1} and some dot products;

[4] Note, however, that for problems that only have small amounts of local interaction, e.g., image denoising, they can perform quite well.

- computing the descent direction as a combination of the preconditioned residual and previous descent direction, $d_k = \tilde{r}_k - \beta_k d_{k-1}$;
- updating the residual using $r_k = r_{k-1} - \alpha_k A d_{k-1}$.

If the residual is updated using this last formula, only one matrix multiply with A needs to be performed per iteration (Step 4 need not be evaluated). The preconditioner matrix M should be a good approximation to A^{-1} and be SPD. It is not necessary to have an explicit expression for M, just the ability to quickly apply M to a vector. More details on this algorithm can be found in (Saad 2003; Shewchuk 1994). Section 4 gives further details on the convergence behavior of PCG.

3.3 Multilevel preconditioners (HBF and ABF)

Multilevel preconditioners (Szeliski 1990, 2006) involve re-writing the original *nodal* variables x as a combination of *hierarchical* variables y that live on a multi-resolution pyramid with L levels. The relationship between x and y can be written as $x = \hat{S}y$, where the reconstruction matrix $\hat{S} = \hat{S}_1 \ldots \hat{S}_L$ consists of recursively interpolating variables from a coarser level and adding in the finer-level variables. The original paper (Szeliski 1990) used a fixed set of full-octave interpolants, while the more recent paper (Szeliski 2006) uses half-octave locally adaptive interpolants, whose values are derived from the structure of the Hessian matrix A (in combination with the sparsification rules discussed in Section 5).

Algorithm 4 Multi-level interpolant precomputation

$[\{S_1 \ldots S_L\}\{A_0 \ldots A_L\}] = \textit{Construct}(A, L)$

INPUT: Hessian matrix A, number of levels L

OUTPUT: Interpolation matrices S_l and coarser level Hessians A_l

$A_0 = A$

1. **for** $l = 1 \ldots L$
2. Select the coarse-level variables x_l
3. Optionally sparsify A_{l-1} (Section 5) //Pre-sparsification
4. Compute the interpolant S_l s.t. $x_{l-1} = S_l x_l$ interpolates the coarse-level variables
5. Apply the Galerkin condition to compute the coarse-level Hessian $A_l = S_l^T A_{l-1} S_l$
6. Optionally sparsify A_l (Section 5) // Post-sparsification
7. **end for**

To construct the bases and the coarse-level Hessian matrices required by this preconditioner, we first call the *Construct* routine given in Algorithm 4, which is also used by multigrid techniques to construct the

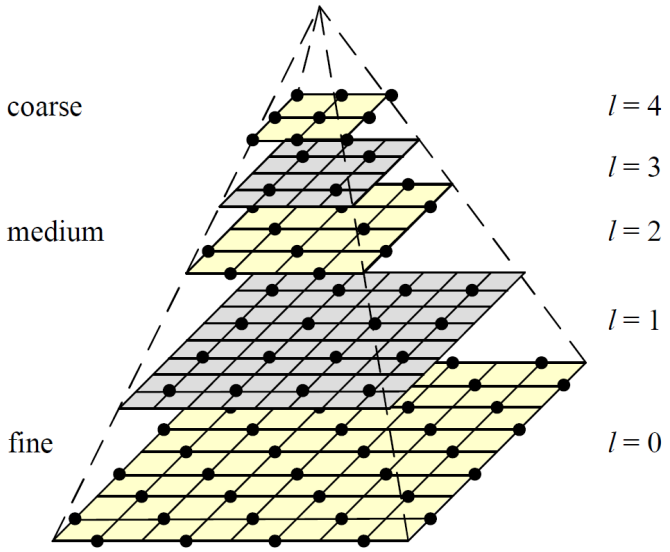


Figure 2: Multilevel pyramid with half-octave sampling (Szeliski 2006).

necessary data structures. Algorithm 4 is written in an iterative form, but it can also be written in a form where it is called recursively to compute higher-level interpolants and coarse-level Hessians. The sparsification steps are used in Algorithm 4 to reduce the number of bands in the coarse level Hessians. If we start with a 5-banded Hessian, the Galerkin condition in Step 5 of Algorithm 4 with a bilinear interpolant results in a 9-banded matrix. To break connections between red and black neighbors and only have connections between red and red neighbors for the next level of half-octave coarsening, sparsification must be performed. See Section 5 for details.

Hierarchical basis preconditioning interpolates coarse-level variables x_l using the interpolation matrix $\hat{S}_l$ and then adds in the remaining fine-level variables $\hat{x}_{l-1}$,

$$x_{l-1} = S_l x_l + \hat{x}_{l-1} = \begin{bmatrix} S_l & 0 \\ & I \end{bmatrix} \begin{bmatrix} x_l \\ \hat{x}_{l-1} \end{bmatrix} = \hat{S}_l \begin{bmatrix} x_l \\ \hat{x}_{l-1} \end{bmatrix} \tag{6}$$

Fig. 3b schematically shows how the hierarchical preconditioner transforms the nodal coordinate system into two subspaces, i.e., a fine level subspace $\hat{x}_{l-1}$, which has axis-aligned contours and hence single-step convergence when adaptive basis functions are used, and a coarse level subspace x_l, which must itself be recursively solved.

The original hierarchical basis paper (Szeliski 1990) uses full octave coarsening to define the coarse level variables in step 2 and bilinear interpolants to define the interpolants in step 4. It then defines the hierarchical basis preconditioner as $M = \hat{S}\hat{S}^T$.

Locally adaptive hierarchical bases (Szeliski 2006) use half-octave (red-black) coarsening in step 2 (Notay and Amar 1997), interpolants derived from the Hessian matrix, and the sparsification techniques described in Section 5. The preconditioner is defined as $M = \hat{S} D_H^{-1} \hat{S}^T$, where D_H is the diagonal matrix formed by concatenating the diagonals of the per-level Hessians A_l. Fig. 2 shows a half-octave pyramid, with the fine-level variables at each level shown as black dots.

In this paper, we directly solve the coarse-level system, as is usually the case in multigrid techniques. Thus, the preconditioner can be written as $M = \hat{S} A_H^{-1} \hat{S}^T$, where

$$A_H = \begin{bmatrix} A_L & & & \\ & D_{L-1} & & \\ & & \ddots & \\ & & & D_0 \end{bmatrix}, \tag{7}$$

and $D_l = \text{diag}(A_l)$ are the per-level Hessian diagonals.

Algorithm 5 is the most general form of the multigrid and multilevel algorithms presented in this paper. Fig. 3 in Section 4 illustrates how the various variants of Algorithm 5 correspond to different search directions and update steps in the multi-dimensional energy landscape. The HBF and ABF algorithms are special cases of this algorithm with $\nu_{pre} = 0$ and $\nu_{post} = 0$. HBF has full-octave coarsening with non-adaptive interpolants; ABF has half-octave coarsening with adaptive interpolants. Depending on the type of sparsification that is performed, (see Algorithm 4 and Section 5), different flavors of ABF arise, which are compared in this paper. For HBF, since we use full-octave coarsening, there is no need for sparsification.

3.4 Geometric multigrid

In this paper, geometric multigrid refers to a multigrid method that uses fixed full-octave coarsening along with fixed non-adaptive (uniform) bilinear interpolation (Trottenberg, Oosterlee, and Schuller 2000). In our experimental results section, we evaluate both Jacobi and Gauss-Seidel four-color (GS-FC) pre-smoothing and post-smoothing operations, and determine that the latter always performs better. The construction phase (Algorithm 4) for geometric multigrid is the same as for regular (non-adaptive) full-octave hierarchical bases.

In this paper, we use geometric multigrid as a preconditioner for PCG, in which case the number of pre-smoothing steps (ν_1) and post-smoothing steps (ν_2) must be the same. In addition, when variants of Gauss-Seidel smoothing, such as raster-order Gauss-Seidel or four-color Gauss-Seidel are used, the ordering of variables must be reversed from pre-smoothing to post-smoothing. The two requirements ensure that the

multigrid preconditioner is symmetric and positive semi-definite (Tatebe 1993). We always use $\nu_1 = \nu_2 = 1$ and refer to this GMG. Algorithm 5 with input flag $d = 0$ then describes GMG (with the appropriate interpolation matrices S_l).

3.5 Algebraic multigrid

Algebraic multigrid (AMG) is a family of multigrid methods that adapt both the coarsening stage and the interpolation function to the structure of the Hessian matrix A. This distinguishes them from the GMG framework, which is non-adaptive in both these aspects. In our current implementation, we use a simplified version of AMG that uses fixed full-octave coarsening. The interpolation operators, however, are adapted to the Hessian using the formulas given in (Trottenberg, Oosterlee, and Schuller 2000, Appendix A). As with GMG, we use Jacobi or four-color pre-smoothing and post-smoothing steps with $\nu_1 = \nu_2 = 1$ and embed AMG into a PCG framework as a preconditioner. The algorithm for AMG preconditioning is therefore Algorithm 5 with differing S_l and A_l matrices and with $d = 0$. Setting $\gamma = 1$ leads to V-cycle preconditioners; setting $\gamma = 2$ leads to W-cycle preconditioners. W-cycles perform more corrections at coarser levels.

Multigrid can also be used as an iterative technique, i.e., a multi-level generalization of Jacobi smoothing. This is equivalent to using preconditioned conjugate gradient (Algorithm 3) with a fixed step size $\alpha_k = 1$ and no conjugation ($\beta = 0$). In this case, we can use a different number of pre- and post-smoothing steps ν_1 and ν_2. We refer to these techniques as V(ν_1, ν_2) in the experimental results section and benchmark both V(1,1) and V(0,1) variants, since setting $\nu_2 = 0$ does not lead to convergent methods (Trottenberg, Oosterlee, and Schuller 2000). V(1, 1) is therefore an iterative (non-PCG) version of AMG in our experiments. These algorithms are described by Algorithm 5 with $\gamma = 1$. There are other variants such as F-cycles (Trottenberg, Oosterlee, and Schuller 2000) which we do not consider. The combinatorial multigrid technique of Koutis *et al.* (2009) uses a W-cycle.

3.6 Combinatorial multigrid

Combinatorial multigrid (CMG) (Koutis and Miller 2008; Koutis, Miller, and Tolliver 2009) refers to a recently developed preconditioner that combines multigrid concepts with tree-based preconditioning ideas (so-called combinatorial preconditioners). The Hessian at each level is represented as an undirected graph A. A second graph B is generated that consists of fewer nodes, such that strongly connected nodes in A are clustered into a single node in B. B can be shown to be a good preconditioner for A while being simpler and therefore easier to invert. The clustering used is a variant of agglomerative clustering that tries to keep fine-level variables strongly connected to their parents, much as in algebraic multigrid. Agglomerative

clustering corresponds to piecewise-constant interpolants, which are faster to apply (both during restriction and prolongation) but have higher aliasing and therefore lead to worse performance on smoother problems. In this paper, we use the code provided by the authors of (Koutis, Miller, and Tolliver 2009) for experimental comparisons.

3.7 A unified multigrid/multilevel algorithm

In Algorithm 5, we describe an algorithm that contains as special cases all of the algorithms discussed above. The input to the algorithm is the current residual r_l, and the output is a correction term e_l, which is added to the current iterate x_l to give the next iterate.

The other inputs to the algorithm are the specific interpolants S_l and Hessians A_l, which depend on the specific algorithm. The interpolants are either data-adaptive (such as AMG, ABF and CMG) or non-adaptive (such as HBF and GMG). The size of the Hessians depends on whether the algorithm contains half-octave coarsening (ABF, HBF), full-octave coarsening (AMG, GMG, V(0, 1) and V(1, 1)), or adaptive coarsening (CMG). As mentioned before, a full implementation of AMG would also involve adaptive coarsening, but we do not have such an implementation available to test.

Line 1 of the algorithm contains an optional pre-smoothing step. Pre-smoothing discovers high-frequency components of the correction. Classic multilevel preconditioners (ABF and HBF) do not perform pre-smoothing. Line 2 of the algorithm finds a new residual, which contains lower frequency components of the correction that are not discovered by pre-smoothing. These lower-frequency components can be efficiently discovered at coarser levels of the grid. To get to the next coarser level, Line 3 applies the transpose of the interpolation operator S_l, to the new residual $\bar{r}_l$.

If we are already at the next-to-coarsest level of the grid, a direct solver is used to solve the system at the coarsest level. Otherwise, a recursive call is made to solve the coarser level system (on Line 10). Depending on the input value of γ, one or more such recursive calls may be made (Lines 8-12). These give rise to the V-cycle or W-cycle correction for $\gamma = 1$ and $\gamma = 2$ respectively. Line 11 gathers together the corrections from one or more such recursive calls. In Line 14, these corrections are interpolated up to the current level. Lines 4 through 14 are common to all multilevel and multigrid algorithms.

An optional diagonal preconditioning of only the fine-level variables at level l is done on Line 16. For a half-octave coarsening, this means that half of the variables are subject to diagonal preconditioning. For a full-octave coarsening, three quarters of the variables are subject to this. Multigrid algorithms do not perform this step, but multilevel algorithms do.

Finally, line 21 has an optional post-smoothing step, which is applied in case of multigrid algorithms.

Algorithm 5 Unified multigrid/multilevel algorithm

$[e_l] = \textit{MGCYC}(l, r_l, \{A_1 \dots A_L\}, \{S_1 \dots S_L\}, L, \omega, \nu^{pre}, \nu^{post}, \gamma, d)$

INPUT: Current level l, residual r_l, per-level Hessians A_l,

per-level interpolation matrices $\hat{S}_l$, number of levels L, damping factor ω,

pre- and post-smoothing iterations ν^{pre} and ν^{post}, number of cycles γ,

flag for optional diagonal preconditioning d

OUTPUT: Correction at level l e_l

1. $e_l^{pre} = \textit{Smooth}\ (0, A_l, r_l, \omega, \nu^{pre})$ // Pre-smoothing correction
2. $\bar{r}_l \ = r_l - A_l e_l^{pre}$ // Update the residual
3. $\bar{r}_{l+1} = S_l^T \bar{r}_l$ // Restrict residual to coarse level
4. **if** $l = L - 1$
5. $\quad e_{l+1} = A_L^{-1} \bar{r}_{l+1}$
6. **else**
7. $\quad e_{l+1} = 0$
8. $\quad$ **for** $j = 1, \dots \gamma$ //$\gamma = 1$ is V-cycle; $\gamma = 2$ is W-cycle
9. $\quad\quad \hat{r}_{l+1} = \bar{r}_{l+1} - A_{l+1} \hat{e}_{l+1}$ // Update the residual
10. $\quad\quad \tilde{e}_{l+1} = \textit{MGCYC}(l+1, \hat{r}_{l+1}, \{A_1 \dots A_L\}, \{S_1 \dots S_L\}, L, \omega, \nu^{pre}, \nu^{post}, \gamma, d)$
11. $\quad\quad e_{l+1} \leftarrow e_{l+1} + \tilde{e}_{l+1}$ // Add up corrections over the cycles
12. $\quad$ **endfor**
13. **endif**
14. $e_l^{cgc} = S_l e_{l+1}$ // Prolong coarse-grid correction
15. **if** (d = 1) // Optional fine-level diagonal preconditioning
16. $\quad e_l^d$ = DiagPrecond($\bar{r}_l$, A_l) // Precondition with inverse diagonal elements of A_l.
17. **else**
18. $\quad e_l^d$ = 0 // No diagonal preconditioning
19. **endif**
20. $e_l^{sum} = e_l^{pre} + e_l^{cgc} + e_l^d$ // Add up corrections: pre-smooth, coarse-level, and diagonal
21. $e_l = \textit{Smooth}(e_l^{sum}, A_l, r_l, \omega, \nu^{post})$ // Post-smoothing

Algorithm	Citation	Smoothing	Adaptive interpolant	Interp. order	Adaptive coarsening	Conjugate gradient
Hier. basis fns.	(Szeliski 1990)	no	no	1	no	yes
Adap. basis fns.	(Szeliski 2006)	no	yes	1	no	yes
Geom. multigrid	(Trottenberg . . . 2000)	yes	no	1	no	yes
Algebraic multigrid	(Trottenberg . . . 2000)	yes	yes	1 or 0	yes	yes
Comb. multigrid	(Koutis . . . 2009)	yes	yes	0	yes	yes
V-cycle	(Trottenberg . . . 2000)	yes	yes	1 or 0	yes	no

Table 1: Summary of multilevel algorithms.

When an algorithm is embedded into a CG framework as a preconditioner, pre-smoothing and post-smoothing algorithms should be the same and the number of iterations ν^{pre} and ν^{post} should also be the same; this ensures that the preconditioning matrix is symmetric and positive definite. Traditional multilevel preconditioners do not have a post-smoothing step. The input to the pre-smoothing step is the sum of the corrections from pre-smoothing, coarse grid solution and diagonal preconditioning. This correction is then modified to compute the final correction.

Looking at Algorithm 5, you can see that it is possible to combine the smoothing elements of multigrid with the fine-level subspace solution (diagonal preconditioning) in multilevel preconditioners. This produces a new algorithm that benefits from both previous approaches, albeit at a higher computational cost. If we set $\nu^{pre} = \nu^{post} = 1$ and $d = 1$, we get a hybrid algorithm that performs pre-smoothing, followed by a coarse-level splitting and preconditioning of the coarse-level and fine-level variables, followed by another step of post-smoothing. This is illustrated in Fig. 3.

In our experimental results Section 7, we show that this hybrid algorithm performs better than either multigrid or multilevel preconditioning used individually. However, since smoothing operations are performed more frequently (at half-octave intervals), it is computationally more expensive than either one of the traditional algorithms. Whether this additional complexity is worth it varies from problem to problem. Generally speaking, once we amortize the convergence rate by the computational cost Section 4, we find that the traditional multilevel approach and hybrid approaches perform comparably.

Table 1 summarizes the multigrid and multilevel preconditioning algorithms discussed in this section. In this table, we use V-cycle to denote an algebraic multigrid algorithm that is not embedded in a CG framework. The interpolation order column refers to the order of the polynomial which may be accurately reconstructed using the interpolation function. Agglomerative clustering has order 0. As you can see, hierarchical basis functions use regular coarsening and no smoothing. Multigrid algorithms use smoothing

operations, with geometric multigrid using regular coarsening and interpolators, and algebraic and combinatorial multigrid using adaptive coarsening and interpolation strategies. Most algebraic multigrid methods use smooth interpolation operators, but some use agglomerative clustering (Grady 2008; Notay 2010; Napov and Notay 2011) like combinatorial multigrid, which corresponds to piecewise constant interpolation.

4 Convergence analysis

To estimate or bound the asymptotic rate at which an iterative algorithm will converge, we can compute the *convergence rate*, which is the expected decrease in error per iteration.

For simple iterative algorithms such as Jacobi and Gauss Seidel (Appendix A), we get a general recurrence of the form

$$x_k = R_k b + H^k x_0 \tag{8}$$

$$= x^* + H^k(x_0 - x^*), \tag{9}$$

where x_0 is the initial solution, x_k is the current solution, $R_k = (I - H^k)A^{-1}$ is derived in Eqn. 37, $x^* = A^{-1}b$ is the optimal (final) solution, and H is the algorithm-dependent *iteration matrix*. The *spectral radius* of the iteration matrix

$$\rho(H) = \max_{\lambda \in \sigma(H)} |\lambda| \tag{10}$$

is the eigenvalue of H that has the largest absolute value.

As the iteration progresses, the component of the error $e = x_0 - x^*$ in the direction of the associated "largest" eigenvector $v_{\max}$ of H, $\tilde{e} = v_{\max}^T e$, gets reduced by a *convergence factor* of ρ at each iteration,

$$\tilde{e}_k = \rho^k \tilde{e}_0. \tag{11}$$

To reduce the error by a factor ϵ (say $\epsilon = 0.1$) from its current value, we require

$$\rho^k < \epsilon \qquad \text{or} \qquad k > \log_\epsilon \rho = -\log_{1/\epsilon} \rho. \tag{12}$$

It is more common (Saad 2003, p.113) to define the *convergence rate* τ using the natural logarithm,

$$\tau = -\ln \rho, \tag{13}$$

which corresponds to how many iterations it takes to reduce the error by a factor $\epsilon = 1/e \approx 0.37$.

The convergence rate allows us to compare the *efficiency* of two alternative algorithms while taking their computational cost into account. We define the *effective convergence rate* of an algorithm as

$$\hat{\tau} = 100\,\tau/C, \tag{14}$$

where C is the amount of computational cost required per iteration.[5] For example, if one algorithm has a convergence rate $\tau = 3$ and another one has a convergence rate of $\tau = 2$ but takes half as much computation per iteration, we can compute the effective rates as

$$\text{First algorithm:} \quad \hat{\tau} = 3/C \tag{15}$$

$$\text{Second algorithm:} \quad \hat{\tau} = 2/(C/2) = 4/C \tag{16}$$

and we see that the second algorithm is more efficient.

For conjugate gradient descent (Shewchuk 1994; Saad 2003), the asymptotic convergence rate is

$$\rho_{CG} \leq \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right), \tag{17}$$

where κ is the condition number of A,

$$\kappa(A) = \frac{\lambda_{\max}}{\lambda_{\min}}, \tag{18}$$

i.e., the ratio of the largest and smallest eigenvalues of A. When the eigenvalues of A are highly clustered, the convergence rate can be even faster (Shewchuk 1994; Saad 2003).

For preconditioned conjugate gradient, the convergence factor in Eqn. 17 depends on the condition number of $B^{-1}A$, where $B = M^{-1}$ is the simple-to-invert approximation of A corresponding to the preconditioner M. The derivations of the formulas for M corresponding to various preconditioners and multigrid techniques are given in Appendix B.

An intuitive way to compute the generalized condition number is to use *generalized Rayleigh quotients*

$$\kappa(B^{-1}A) = \kappa(A, B) \equiv \max_x \frac{x^T A x}{x^T B x} \cdot \max_x \frac{x^T B x}{x^T A x} \tag{19}$$

(Koutis, Miller, and Tolliver 2009). Each of the quadratic forms $x^T Ax$ can be interpreted as the power dissipated by network A, where the edge weights a_{ij} give the conductances and the vector x specifies the input voltages (Fig. 1).

Thus, a good preconditioner B is one that dissipates neither significantly more nor significantly less power than the original matrix A across all possible voltages (both smooth and non-smooth). We use this observation to develop better sparsification rules in the next section.

For HBF and ABF, the condition number κ can be computed using Eqn. 19 using the inverse of the preconditioner M as the approximation to the Hessian $B = M^{-1}$. The formulas for M are explicitly given

[5] To make these numbers more similar across problems and independent of grid size, we define C as the number of floating point operations per grid point in the fine grid. The scale factor of 100 makes $\hat{\tau}$ easier to print and corresponds roughly to how many flops it takes to perform one multigrid cycle.

on p. 13, i.e., $M = SS^T$ for HBF and $M = SA_H^{-1}S^T$ for ABF. For V-cycle multigrid, we use instead the spectral radius of the preconditioned system to derive theoretical convergence rates. However, in practise, instead of using these formulae, we derive the condition numbers κ or the spectral radius (Eqn. 10) using power iterations to derive the largest and smallest eigenvalues.

To summarize, the convergence factor ρ for a standard iterative algorithm such as Jacobi, Gauss-Seidel, or multigrid, depends on the spectral radius of the associated iteration matrix H (Eqn. 10). For preconditioned conjugate gradient, it depends on the generalized condition number $\kappa(A, B)$, which can be thought of as the ratio of the greatest increase and decrease in relative power dissipation of the preconditioner matrix B with respect to the original matrix A. In our experimental results section, we compute the theoretical convergence factor ρ, the convergence rate τ, and the effective convergence rate $\hat{\tau}$ for problems that are sufficiently small to permit easy computation of their eigenvalues. For all problems, we also estimate an *empirical* convergence rate $\tilde{\tau}$ by dividing the logarithmic decrease in the error $|e_k|$ by the number of iterations,

$$\tilde{\tau} = -\frac{1}{N} \ln \frac{|e_N|}{|e_0|}. \tag{20}$$

4.1 Computational complexity

To estimate the computational complexity C of a given algorithm, we count the number of floating point operations (multiplies, adds, subtractions, divisions, and square roots) used at each iteration. Roughly speaking, multiplication by the sparse Hessian matrix A takes about $2b$ flops per pixel,[6] where $b \in \{5, 9\}$ is the size (bandwidth) of the stencil. In practice, we add code to our MATLAB implementation to count the number of non-zeros in A and add twice this number to our flop count. Similarly, for adding, subtracting, multiplying vectors by scalars or diagonals, we add one flop per pixel, and count dot products as two.

The bulk of the effort in a preconditioned conjugate gradient technique is split between the computation and update of the residual, which takes about $2b + 1$ operations, the computation of the step size α and conjugation amount β (2-4 flops each), the update of the state and residual vectors (4 flops), and the multilevel smoothing or preconditioning.

The latter depends on the size of the interpolation filter, the number of smoothing steps, and the amount of work performed in the coarse-level solve. The interpolation filters for full-octave techniques such as hierarchical basis functions (HBF), geometric multigrid (GMG), and our full-octave implementation of algebraic multigrid (AMG) takes on average just over 4 flops per pixel each for restriction and prolongation, with a corresponding 1/4 amount for each subsequent level. For half-octave adaptive basis functions, the

[6] Here, we use the term "pixel" to denote one variable in the fine-level solution.

amount of work is about about 8 flops per pixel, since half the grid at any time is being interpolated from four parents. Combinatorial multigrid has 1/4 the computational cost of GMG, since each pixel belongs to a single cluster, and only 1 addition is needed for restriction or prolongation.

For the coarse-level solve, we count the number of non-zeros in the Cholesky factorization of the coarsest level Hessian A_c and multiply this by 4 (one multiply-add each for forward and backward substitution). For regular 2D grids reordered using nested dissection, the number of nonzeros in the upper-triangular factor matrix is roughly $2n \log_2 n + O(n)$ (Davis 2006, p.130), where n is the number of variables in a square grid.[7] In our MATLAB implementation, we use the AMD (approximate minimum degree) re-ordering heuristic built into the Cholesky decomposition routine to count the number of non-zeros (or the Cholesky factorization produced by the CMG code).[8]

The amount of effort required for smoothing using Jacobi or red-black or four-color Gauss Seidel is similar to that required for residual computation or update. After the initial smoothing step(s), a new residual needs to be computed (Step 3 in Algorithm 5). Thus, roughly speaking, symmetric multigrid preconditioners or multigrid solvers require four times more work ($8b + 4$ vs. $2b + 1$) per level than hierarchical basis preconditioners, ignoring the cost of inter-level transfers and the coarse-level solve.

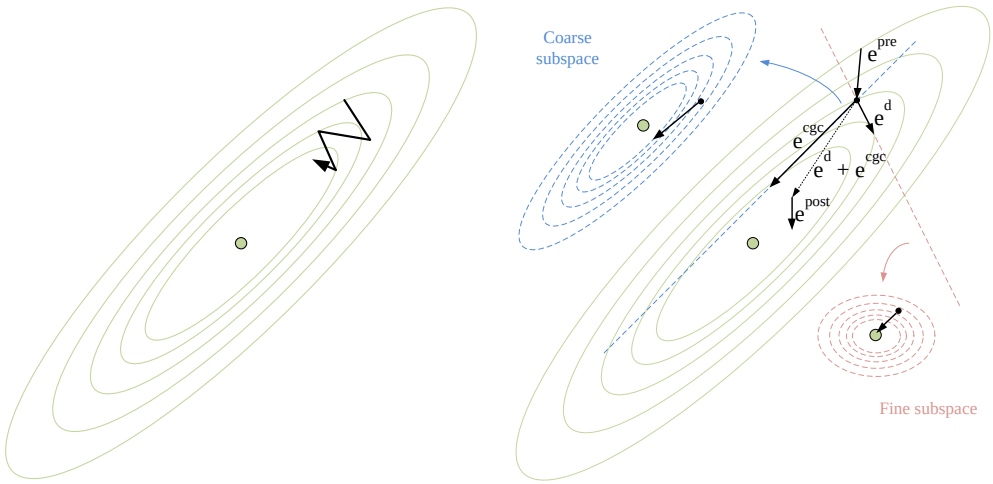
Putting all of these results together, and assuming that the amount of work involved in inter-level interpolation is comparable across techniques and similar to that involved in a smoothing step, we expect the hierarchical basis techniques (HBF and ABF) to be comparable in computational cost to CMG and about half the cost of symmetric multigrid preconditioners (AMG and GMG). The exact numbers we observe are given in our experimental results section. However, as we note in our discussion section, we should remember that flop counts are themselves not a complete predictor of actual performance, since issues such as caching and memory access patterns will be important in high-performance implementations of these algorithms.

5 Sparsification

As we described in Section 3.3, the adaptive hierarchical basis function algorithm of (Szeliski 2006) relies on removing unwanted "diagonal" links after each round of red-black multi-elimination, as shown in Fig. 4.

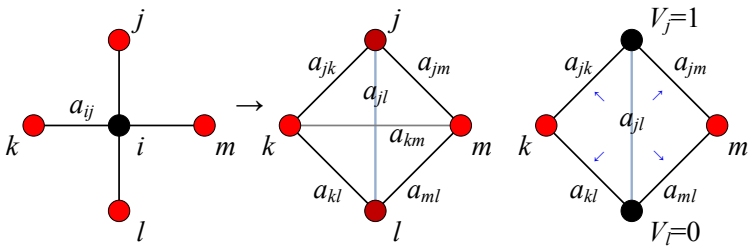
[7] The cost of computing the original factorization can be significantly higher, about $10n^{3/2}$ flops, but we do not currently measure setup costs.

[8] In some experiments, we found that the AMD re-ordering produces slightly sparser factorization than nested dissection for $\mathcal{N}_4$ grids but slightly worse of $\mathcal{N}_8$ grids, with the difference being less than 10%.



(a) (b)

Figure 3: Energy contours and descent steps for the various iterative algorithms. (a) Steepest descent, e.g., Jacobi or Gauss-Seidel smoothing. (b) Multigrid/multilevel preconditioning: an optional pre-smoothing correction step (e^{pre}); followed by a coarse-level solve (e^{cgc}); optional diagonal pre-conditioning (e^{d}); followed by another optional post-smoothing step (e^{post}). In multilevel algorithms, the fine subspace solution is exact. For all algorithms, the coarse subspace solution is approximate except at the coarsest level.



(a) (b) (c)

Figure 4: Sparsification: (a) after the black node i is eliminated, the extra "diagonal" links a_{jl} and a_{km} shown in (b) are introduced. (c) Since nodes j and l (now shown in black) are eliminated at the next round, only the a_{jl} edge needs to be eliminated and redistributed to the adjacent edges a_{jk}, a_{kl}, a_{jm}, and a_{ml}.

The rule for re-distributing an unwanted connection a_{jl} to its neighbors is

$$a'_{jk} \leftarrow a_{jk} + s_{\mathrm{N}}\, a_{jk} a_{jl} / S, \tag{21}$$

where $S = a_{jk} + a_{kl} + a_{jm} + a_{lm}$ is the sum of the edge weights adjacent to a_{jl} and $s_{\mathrm{N}} = 2$ is a constant chosen to reproduce the finite element discretization at the coarser level. The diagonal entries corresponding to the edges being eliminated and those being "fattened" are modified appropriately to maintain the same row sums as before, thereby resulting in the same energy for constant-valued solutions.[9]

In this paper, we introduce several extensions and improvements to this original formulation:

$\mathcal{N}_8$ grid sparsification. Although the original paper only described how to handle 4-neighbor (5-point stencil) discretizations, it is straightforward to apply the same sparsification rule used to eliminate unwanted diagonal elements from an 8-neighbor (9-point stencil) discretization. We demonstrate this in our experiments with colorization algorithms.

Fine-fine sparsification. The original algorithm eliminates both sets of "diagonal links", i.e., a_{jl} and a_{km} in Fig. 4b. Closer inspection reveals that this is unnecessary. At the next level of coarsening, only two out of the four nodes, say j and l (shown as dark red nodes in Fig. 4b and black in Fig. 4c), will be themselves eliminated. Therefore, it is only necessary to sparsify the links connecting such nodes, which results in a lower approximation error between the original and sparsified network. The other a_{km} link is left untouched and does not participate in the computation of the adaptive interpolants or cause the coarse-level Hessian matrix to grow in bandwidth. In the subsequent text, we refer to this sparsification as the "7-point stencil" sparsification, since the Hessian matrix after sparsification is on average 7-banded. (Red nodes are connected to 8 neighbors, while black or fine nodes, which are about to be eliminated, are only connected to 4 parents—see Fig. 5 for a small numerical example.)

Pre-sparsification. The original algorithm in Szeliski (2006) performs sparsification (elimination of "diagonal links") each time a coarse-level Hessian is computed using the Galerkin condition (Szeliski 2006, Eqn. (13)). However, unless the system is going to be further coarsened, i.e., unless another set of adaptive interpolants needs to be computed, this is unnecessary. The higher-bandwidth coarse-level matrix can be used as-is in the direct solver, at a slight increase in computational cost but potentially significant increase in accuracy (preconditioning efficacy). We call deferring the sparsification step to just before the interpolant construction as *pre-sparsification* and call the original approach *post-sparsification*. Both variants are evaluated in our experimental results section.

[9] The modified incomplete LU (Cholesky) or MILU algorithm also maintains row sums, but it simply drops off-diagonal entries instead of re-distributing them to adjacent edges (Saad 2003; Szeliski 2006).

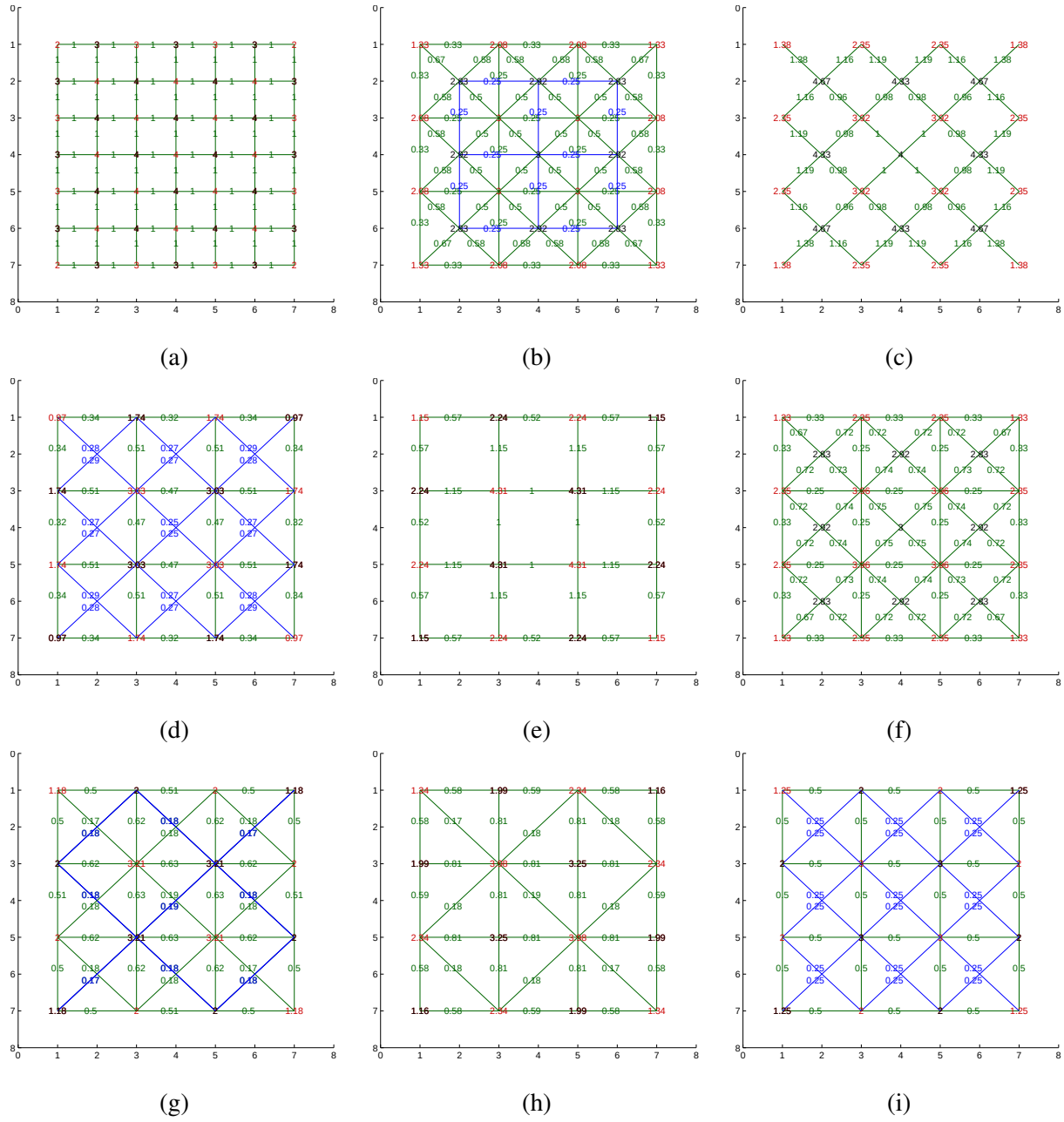


Figure 5: Numerical example of sparsification: (a) original uniform Laplacian grid; (b) after half-octave elimination of the black (fine) variables; (c) after sparsification to a 5-banded structure (elimination of "diagonal" links); (d) second half-octave elimination; (e) sparsification to a full-octave 5-banded coarsening; (f) partial sparsification to a "7-band" structure (red nodes have 8 neighbors, black nodes, which are about to be eliminated, have 4 parents); (g) second half-octave elimination; (h) full-octave coarsening; (i) AMG full-octave coarsening.

Maintaining positive definiteness and diagonal dominance. One question that might arise is whether the sparsification rules we use can result in a Hessian that was originally positive definite becoming indefinite or negative definite. Fortunately, if we use the locally adapted interpolants derived in (Szeliski 2006), we can prove that it does not. The general Galerkin condition rule for computing a coarse-level matrix A_c from a fine-level matrix A_f is $A_c = R^T A_f R$, where R is the interpolation matrix; it is straightforward to show that if A_f is SPD, then A_c must be as well. For locally adapted hierarchical bases constructed using the rule in (Szeliski 2006, Eqn. (12)),

$$R = [-D^{-1}E \quad I], \tag{22}$$

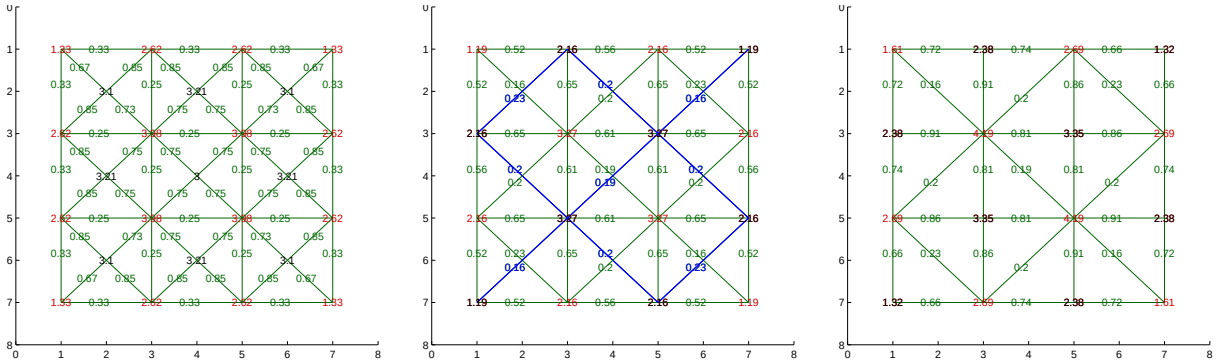
where D is the diagonal of the fine-level Hessian and E are the off-diagonal elements of A_f, the formula for A_c becomes (Szeliski 2006, Eqn. (13))

$$A_c = F - E^T D^{-1} E. \tag{23}$$

Thus, if we start with a fine-level Hessian where all the off-diagonals in E are negative, which is the case for problems defined using pairwise quadratic energies such as Eqn. 1 and Eqn. 4, we end up with a coarse-level Hessian A_c also has negative off-diagonals and can therefore be viewed as a resistive grid. The sparsification rules developed in (Szeliski 2006) and extended in this paper ensure that if the original a_{jl} values are all negative (corresponding to positive resistances) and the matrix is originally SDD, the sparsified matrix will be as well.

Energy-based sparsification. The original adaptive basis function paper (Szeliski 2006) re-distributes edge weights in proportion to the relative strengths of the remaining edges (Eqn. 21). As we discussed in Section 4, a good preconditioner B has a small generalized condition number with respect to the original Hessian A, which in turn depends on the generalized Rayleigh quotients in Eqn. 19. Thus, we can view sparsification as the process of adapting (strengthening) the remaining links in Fig. 4c when the unwanted link a_{jl} is eliminated.

Consider the energy dissipated by the resistive network shown in Fig. 4c, remembering that the edge values such as a_{jl} correspond to conductances. This kind of voltage configuration, which is dissipating a smooth gradient from top to bottom (we let the voltages at nodes k and m float), is typical of the low-frequency modes whose power dissipation we are trying to preserve after sparsification. The central link, which is being eliminated, dissipates $a_{jl}(1)^2 = a_{jl}$. The (series) conductances on the left and right branches are $\hat{a}_{jkl} = (a_{jk}^{-1} + a_{kl}^{-1})^{-1}$ and $\hat{a}_{jml} = (a_{jm}^{-1} + a_{ml}^{-1})^{-1}$ and so are their power dissipations. If we drop the



(a) (b) (c)

Figure 6: Energy-based sparsification. Compare the values in these grid to those in Figure Fig. 5f–h. We see that they are comparable, although small differences exist near some of the edges.

central link and beef up all the other links by a factor s,

$$a'_{jk} \leftarrow a_{jk} + s a_{jk}, \tag{24}$$

we need to satisfy the following equation to obtain a Rayleigh quotient of 1:

$$a_{jl} + \hat{a}_{jkl} + \hat{a}_{jml} = (1+s)(\hat{a}_{jkl} + \hat{a}_{jml}) \tag{25}$$

or

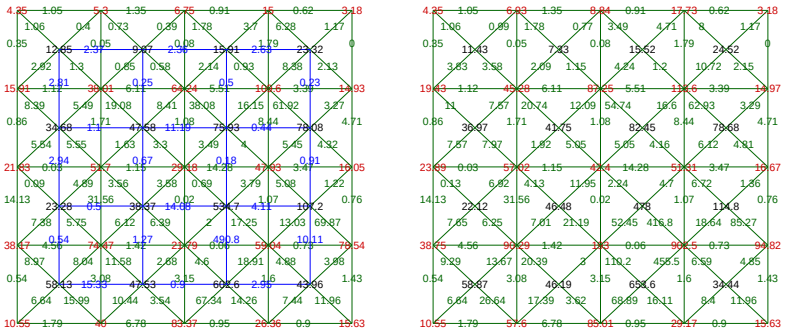
$$s = \frac{a_{jl}}{\hat{a}_{jkl} + \hat{a}_{jml}}. \tag{26}$$

While in the original paper (Szeliski 2006) the edge increment values (Eqn. 21) derived from all eliminated edges are added together, it makes more sense in an energy-based analysis to take the maximum value of s computed for each edge being fattened with respect to all the eliminated edges it compensates for.

How does this energy-based approach compare to the original formulation? Consider the case of a uniform grid, where $a_{jk} = a_{kl}$ and $a_{jm} = a_{ml}$. Hence, we have $\hat{a}_{jkl} = (a_{jk}+a_{kl})/4$, $\hat{a}_{jml} = (a_{jm}+a_{ml})/4$, and the denominator D in Eqn. 26 is $D = S/4$, where S is the sum of all the side conductances in Eqn. 21. Now, equating Eqn. 21 with Eqn. 24 using Eqn. 26 and $D = S/4$, and remembering that each edge being fattened in Eqn. 21 gets contributions from two different sides, we see that the choice of $s_N = 2$ in in Eqn. 21 leads to comparable results in the case of a uniformly weighted grid.

Fig. 6 shows the results of applying energy-based sparsification to the same problem originally shown in Fig. 5f–h. As you can see, the results are quite similar, although small some small differences can be seen.

The limits of sparsification. Given that our sparsification rules are based on a sensible approximation of the low-frequency modes of one matrix with a sparser version, we can ask if this always result in a



(a) (b)

Figure 7: Limits of sparsification: (a) grid for a highly irregular problem before sparsification and (b) after partial sparsification. Note how the edge labeled 490.8 around (6,7) is distributed to its four much smaller neighbors, causing a large generalized condition number

reasonably good approximation of the original problem. Unfortunately, this is not always the case. Consider the case where the value of $|a_{jl}|$ is much larger than all of the adjacent edge values in Fig. 4c. In this case, the sparsification rule (Eqn. 21) can increase the values of the adjacent edges by an arbitrarily large ratio $s_{\mathrm{N}} a_{jl}/S \gg 1$. The problem with this is that while the power dissipation of the sparsified system to the 0/1 voltage shown in Fig. 4c can be kept constant, if we then set all but one (say $V_j = 1$) of the voltages in our network to zero, the power dissipated by the original and sparsified network can have an arbitrarily large Rayleigh quotient (Eqn. 19) and hence an arbitrarily bad generalized condition number.

A more intuitive way of describing this problem is to imagine a grid where neighbors are connected in long parallel chains (say a tight spiral structure, or two interleaved spirals with strongly differing values or voltages). If we use regular two-dimensional coarsening, there is no way to represent a reasonable approximation to this distribution over a coarser grid. Adaptive coarsening schemes such as CMG or full AMG, on the other hand, have no trouble determining that a good coarsening strategy is to drop every other element along the long linear chains. Therefore, an approach that combines sparsification rules with adaptive coarsening may perform better for such problems, and is an interesting area for future work.

6 Sample problems

To measure the performance of the various techniques, we analyze a number of quadratic minimization problems that arise from the solution of two-dimensional computer graphics and computational photography problems.

The problems in this section are ordered from most regular to most irregular, which corresponds roughly to the level of difficulty in solving them. Regular problems have spatially uniform (homogeneous) data ($w_{i,j}$) and smoothness ($s_{i,j}$) weights and are well suited for traditional multigrid techniques. Moderately regular problems may have irregular data constraints and occasional tears or discontinuities in the solution. Anisotropic (directionally selective) smoothness terms also fall into this category and often require adaptive subsampling techniques. The most challenging problems are those where the smoothness can be very irregular and where elongated one-dimensional structures may arise. Techniques based on regular coarsening strategies generally do not fare well on such problems.

6.1 High Dynamic Range (HDR) Image Compression

HDR image compression (Fattal, Lischinski, and Werman 2002; Lischinski, Farbman, Uyttendaele *et al.* 2006) (also known as tone mapping) maps an input image whose values have a high dynamic range into an 8-bit output suitable for display on a monitor. The gradients of the log of the input luminance are compressed in a non-linear manner. An image is then reconstructed from the compressed gradients $\{g^x_{i,j}, g^y_{i,j}\}$ by minimizing Eqn. 1. We follow the formulation given in (Fattal, Lischinski, and Werman 2002) to perform the HDR compression. Fig. 8 shows the two images used for our HDR compression tests.

6.2 Poisson Blending

In Poisson blending (Pérez, Gangnet, and Blake 2003; Levin, Zomet, Peleg *et al.* 2004; Agarwala *et al.* 2004), gradient domain operations are used to reduce visible discontinuities across seams when stitching together images with different exposures. There are a few different variants of Poisson blending. In our implementation, we blend each of the RGB channels separately, using the formulas provided in Section 2 of (Szeliski, Uyttendaele, and Steedly 2011). Fig. 9 gives two examples of Poisson blending, where two input images of the same scene are stitched together along a seam. The seam is almost unnoticeable because the stitching is done in the gradient domain.

6.3 2D membrane

A membrane refers to an energy-minimizing 2D piecewise smooth function used to interpolate a set of discrete data points (Terzopoulos 1986). It is a good canonical problem to test the performance of techniques for solving spatially-varying (inhomogeneous) SPD linear systems. Our current test problem, modeled after those previously used in (Szeliski 1990, 2006), uses four data points at different heights and a single tear

(a) (b)

(c) (d)

Figure 8: Two examples of High Dynamic Range compression (Fattal, Lischinski, and Werman 2002): "Belgium" (a) Input and (b) Compressed; "Memorial" (c) Input and (d) Compressed.

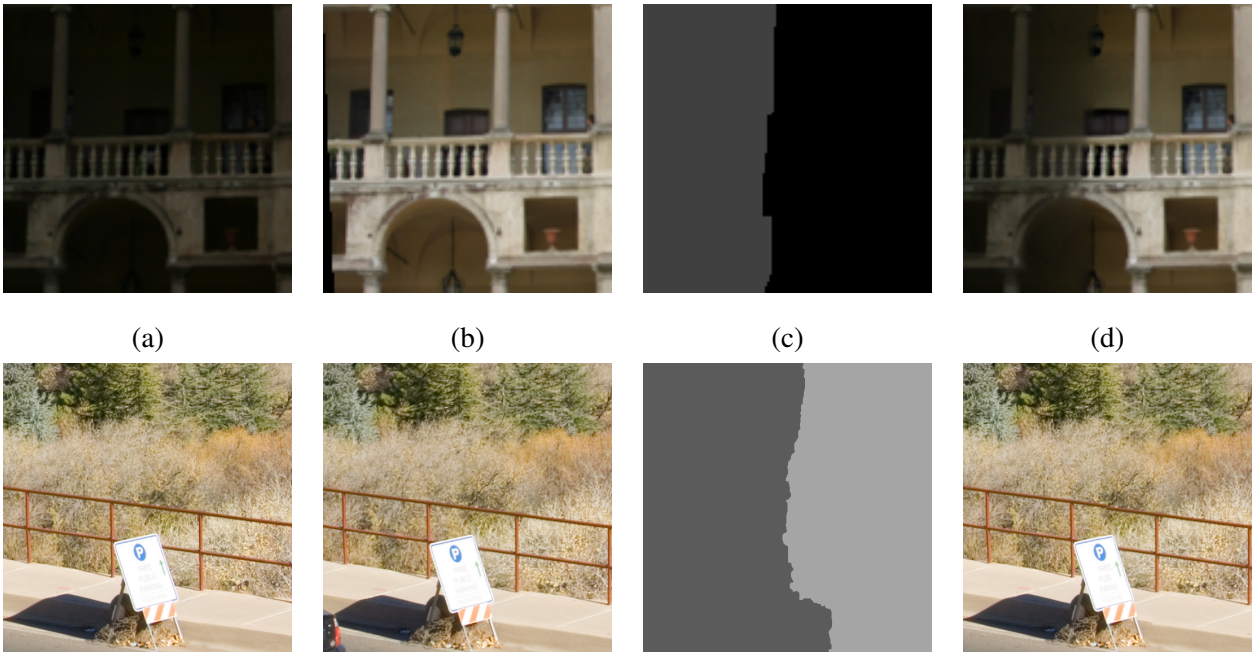


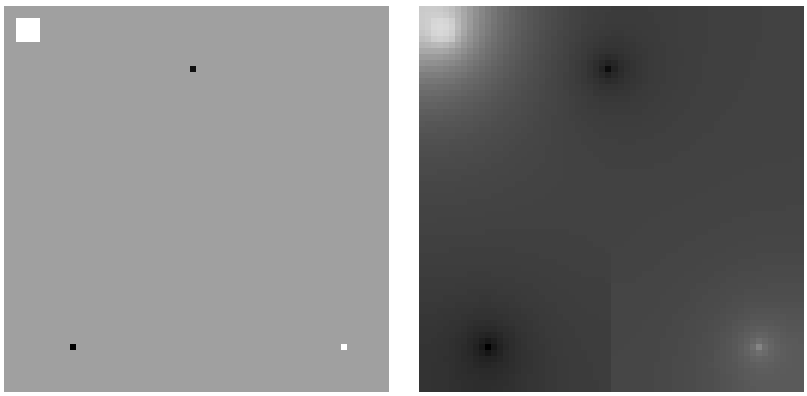
Figure 9: Two examples of Poisson blending: (a,e) first source image; (b,f) second source image; (c,g) labels; (d,h) blended result.

(zeros in the $s^x_{i,j}$ weights at appropriate locations) running halfway up the middle of the surface (Fig. 10).

6.4 Colorization

In colorization (Levin, Lischinski, and Weiss 2004), a gray-scale image is converted to color by propagating user-defined color strokes. In order to prevent color bleeding across edges, the propagation of the color is controlled by the strength of edges in the gray-scale image. Let Y be the gray-scale luminance image and S the stroke image. As in (Levin, Lischinski, and Weiss 2004), we work in YIQ color space. giving the following data weight and gradient weight terms:

$$\begin{aligned} w_{i,j} &= 100 \quad \text{if user has drawn a stroke at} \quad (i,j) \\ w_{i,j} &= 0 \quad \text{if user has not drawn a stroke at} \quad (i,j) \end{aligned}$$



(a) (b)

Figure 10: An example of 2D membrane interpolation (Szeliski 2006): (a) input image with data points and a tear down the middle bottom (not visible) (b) interpolated result.

$$
\begin{aligned}
d_{i,j} &= \text{color of the stroke} \\
s^x_{i,j} &= \frac{1}{|1 + 0.2 * (G_{i+1,j} - G_{i,j})^2|} \\
s^y_{i,j} &= \frac{1}{|1 + 0.2 * (G_{i,j+1} - G_{i,j})^2|} \\
s^{xy}_{i,j} &= \frac{1}{|1 + 0.2 * (G_{i+1,j-1} - G_{i,j})^2|} \\
s^{yx}_{i,j} &= \frac{1}{|1 + 0.2 * (G_{i+1,j+1} - G_{i,j})^2|}
\end{aligned}
$$

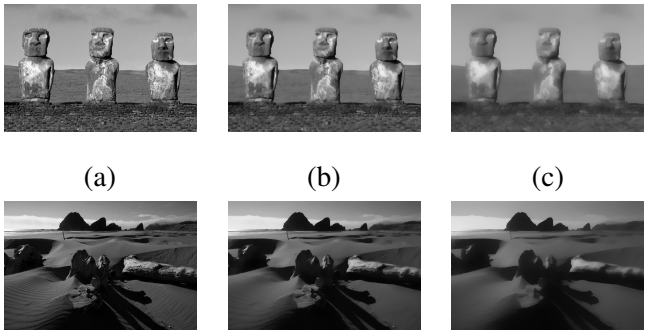
and all the gradient terms g^x, g^y, g^{xy}, g^{yx} are set to 0. The system Eqn. 2 or Eqn. 4 is solved to recover both the I and Q color channels. This is combined with Y to give the output YIQ image. If s^{xy} and s^{yx} are set to 0, a 5-band system results, otherwise we have a 9-band system. Fig. 11 gives an example of colorization.

6.5 Edge-preserving decomposition (EPD)

An edge-preserving multi-scale decomposition of an image can be created using non-linear filters to smooth an image while preserving sharp edges (Farbman, Fattal, Lischinski *et al.* 2008). The non-linear smoothing step is performed by minimizing a system of equations similar to those used in colorization, except that instead of sparse color strokes, the complete input image is used as a weak constraint on the final solution, i.e., $w_{i,j}$ is small but constant everywhere. We use this as another example of an inhomogeneous Poisson reconstruction problem, since the smoothness weights $s_{i,j}$ are spatially varying. Fig. 12 gives an example of edge-preserving decomposition, with the original image being passed through 2 successive layers of edge-preserving smoothing.

(a) (b) (c) (d) (e)

Figure 11: Colorization using a variety of multilevel techniques: (a) input gray image with color strokes overlaid; (b) "gold" (final) solution; (c) after one iteration of adaptive basis functions with partial sparsification; (d) after one iteration of regular coarsening algebraic multigrid with Jacobi smoothing and (e) with four-color smoothing. The computational cost for the ABF approach is less than half of the AMG variants and the error from the gold solution is lower (zoom in to see the differences). Four-color smoothing provides faster convergence than Jacobi.



(a) (b) (c)

(d) (e) (f)

Figure 12: An example of edge-preserving decomposition (Farbman, Fattal, Lischinski *et al.* 2008): (a) input gray-scale image; (b) first layer of edge-preserving smoothing; (c) second layer of edge-preserving smoothing (by smoothing first layer). (d)-(f) input gray-scale image and two layers of smoothing for another gray-scale image;

	HDR			Poisson			Membrane		Color. 5-pt			Color. 9-pt			EPD		
	33	512	768	33	128	512	33	256	32	256	239	32	256	239	33	800	535
	(4)	(8)	(7)	(4)	(6)	(7)	(4)	(7)	(4)	(7)	(6)	(4)	(7)	(6)	(4)	(8)	(7)
HBF	3.35	2.09	2.78	3.54	2.80	2.41	0.94	0.7	0.92	1.18	0.93	0.96	1.06	0.82	1.97	1.78	1.37
ABF-Pre7	5.42	2.81	2.88	5.39	4.48	4.83	**6.41**	**5.19**	8.75	**9.97**	**5.65**	4.55	**4.17**	**3.68**	3.82	3.09	2.54
ABF-Pre7-W	3.15	2.42	2.56	3.41	2.86	2.42	3.98	3.15	2.72	3.43	0.39	1.22	0.65	0.43	0.57	0.25	0.26
ABF-Pre7-4C	7.15	**5.55**	4.75	8.01	**7.77**	**8.49**	4.64	4.29	**11.29**	5.90	3.28	2.65	2.85	2.18	4.03	2.23	1.93
ABF-Pre7-4C-W	2.55	1.69	1.91	2.41	1.80	1.74	1.44	0.77	2.51	1.03	0.88	0.71	0.44	0.39	1.14	0.14	0.15
AMG-J	4.37	3.20	3.73	4.82	4.40	4.46	3.99	3.32	2.77	2.23	1.10	3.13	1.83	1.07	1.47	0.93	0.72
AMG-4C	**7.53**	4.56	4.95	**8.25**	6.73	7.21	5.35	4.29	5.58	3.30	1.68	4.02	2.75	1.17	1.91	1.19	0.92
AMG-4C-W	4.66	5.32	**5.16**	5.21	5.25	5.19	3.94	3.30	3.74	2.95	1.81	3.34	2.68	1.70	1.57	1.28	1.16
GMG-J	4.65	3.48	4.17	5.09	4.41	4.41	1.06	0.33	1.39	1.14	0.78	1.87	1.21	0.83	0.99	0.89	0.71
V(1,1)-4C	6.95	4.59	4.93	7.67	5.03	6.34	2.93	3.68	5.76	2.93	1.43	3.97	2.22	1.00	0.93	0.82	0.39
V(0,1)-4C	4.74	2.74	4.28	6.30	3.96	5.36	5.49	5.12	4.68	3.66	2.52	**5.09**	2.76	1.55	1.48	1.35	0.62
CMG	1.36	1.16	1.25	1.92	1.87	2.18	1.82	1.97	-	2.82	2.39	1.56	1.81	1.89	**4.87**	**4.44**	**4.24**

Table 2: Empirical effective convergence rates $\hat{\tau}$ for preconditioners applied to all of our sample problems. The **boldface** numbers are the "winners" (fastest empirical convergence rate) in each column. For each problem, test results on two image sets and a small version of one of the sets are provided. The number of rows in the test image is at the top of each column. Below that, the number of levels is given in parentheses.

7 Experiments

In this section, we compare the solvers of Section 3 on the different problems outlined in Section 6.

Appendix E lists all of our experimental results, both in tabular form for the convergence rates and as plots of error vs. both iteration count and computational cost (100s of flops). For problems that are sufficiently small, the theoretical convergence rates based on the generalized condition numbers of the preconditioners or the spectral radius of the iteration matrix are also given. Table 2 summarizes the empirical convergence rate $\hat{\tau}$ using three levels of preconditioning applied to all of our sample problems, with the best performing algorithm for each problem shown in boldface.

In these tables and figures, the algorithms are denoted as follows:

- HBF: full-octave non-adaptive hierarchical basis preconditioning (Szeliski 1990);
- ABF-Pre7: half-octave adaptive hierarchical basis preconditioning (Szeliski 2006), where sparsification is applied before the interpolants and coarse-level Hessians are computed,
- ABF-Pre7-W: a W-cycle variant of ABF-Pre7;
- ABF-Pre7-4C: ABF-Pre7 with four-color Gauss-Seidel pre-smoothing and post-smoothing; this is the unified algorithm described in Appendix D with all components (smoothing, fine-level diagonal preconditioning and coarse-level solve) enabled;

	HDR	Poisson	Membrane	Color. 5-pt	Color. 9-pt	EPD
	2048×1365	2048×2048	2048×2048	1854×2048	1854×2048	1365×2048
	(10)	(10)	(10)	(8)	(8)	(8)
HBF	2.23	2.58	0.96	0.41	0.56	1.65
ABF-Pre7	2.61	5.52	3.97	**20.35**	**16.48**	2.60
ABF-Pre7-4C	**5.75**	**9.52**	3.00	12.17	7.56	1.74
AMG-J	2.93	5.15	3.35	3.00	2.71	0.60
AMG-4C	4.16	7.69	3.33	8.62	4.16	0.87
GMG-J	3.21	4.98	0.57	1.05	0.98	0.91
V(1,1)-4C	4.06	6.72	3.45	8.94	4.29	0.37
V(0,1)-4C	3.55	5.48	**4.02**	1.41	2.42	0.59
CMG	-	2.29	1.75	5.58	3.55	**4.18**

Table 3: Empirical effective convergence rates $\hat{\tau}$ for preconditioners applied to all of our sample problems for multi-megapixel images. Next to problem size, the number of levels is given in parentheses. The **boldface** numbers are the "winners" (fastest empirical convergence rate) in each column. The size of the tested image is at the top of each column. Below that is the number of levels in parentheses.

- ABF-Pre7-4C-W: W-cycle variant of ABF-Pre7-4C;
- AMG-S: algebraic multigrid preconditioning *with a fixed full-octave coarsening scheme* and either a Jacobi (J) smoother or four-color Gauss-Seidel (4C) smoother; (our current implementation does not do full justice to AMG, since we were unable to find adaptive coarsening code);
- GMG-J: geometric multigrid preconditioning with full-octave bilinear interpolants and Jacobi smoothing;
- V(m,n)-4C: V-cycle multigrid with a fixed step size and four-color Gauss-Seidel m pre-smoothing and n post-smoothing steps; the interpolants and Hessians are computed using the same logic as for AMG-S; this variant tests the effectiveness of using stand-along multigrid cycles instead of PCG;
- CMG: combinatorial multigrid, using the code provided by the authors of (Koutis, Miller, and Tolliver 2009).

There are other possible variants such as the original ABF 5-banded sparsification algorithm from (Szeliski 2006). The new partial sparsification we introduce always works better than the original sparsification both empirically and theoretically, so we do not include these runs in our results.

Looking at the complete set of results available in Appendix E, we see that the relative behavior of the various algorithms depends both on the amount of smoothness and the amount of inhomogeneity in the sample problems, i.e., how regular they are.

HDR and Poisson blending are homogeneous problems, ideally suited to geometric multigrid. Indeed

GMG (and AMG, which reverts to GMG on uniform problems) have very fast convergence rate τ. Furthermore, τ is independent of the number of levels, as is predicted by multigrid theory. However, when we factor in the computational cost C, the effective convergence rate $\hat{\tau}$ for the ABF-Pre7-4C is significantly better. Using the new partial (7-point stencil) sparsification rule consistently lowers the condition numbers and usually helps the empirical performance. Comparing the empirical rates for AMG-J and AMG-4C shows that the four-color smoother is a much better choice than Jacobi smoothing for all problems, but has an especially dramatic effect for the homogeneous problems.

The 2D membrane exhibits a moderate amount of inhomogeneity, including scattered data constraints and a strong discontinuity. Here, non-adaptive techniques such as HBF and GMG-J start to fall apart. ABF-Pre7 starts to pull away, with the additional smoothing of ABF-Pre7-4C not providing significant improvement. AMG-4C is still competitive if we ignore computational costs. CMG, even though it's designed for irregular problems, has slower convergence because it uses a piecewise-constant interpolator for what is essentially a mostly smooth problem.

The results on colorization (for both 5-point and 9-point fine-level stencil formulations) are similar to the 2D membrane, even though the number of discontinuities is quite a bit larger. Here, we would expect a full (adaptive coarsening) implementation of AMG-4C to do better, but we currently don't have such an implementation to test. In terms of theoretical effective convergence rates, CMG is starting to look competitive, but still doesn't do as well empirically.

The edge-preserving decomposition is much more challenging. Because of the large number of discontinuities and thinner elongated structures, CMG starts to perform very well, both in terms of theoretical condition numbers, and in terms of practical convergence rates. It will be interesting to compare its performance against true (adaptive coarsening) algebraic multigrid, and also to future extensions of our sparsifying approach that use adaptive coarsening schemes.

Note that for these problems, the theoretical convergence rates predicted by condition number analysis are much worse than the actual empirical convergence rates we observe. We believe the reason for this is the clustering of eigenvalues after preconditioning, whereas the theoretical condition number is computed using the largest and smallest eigenvalues, which may be outliers.

The unified algorithm ABF-Pre7-4C performs well on all the problems, even after taking into account computational load. Further research into combining adaptive coarsening with the unified algorithm may result in better performance for the less smooth problems such as edge-preserving decomposition.

The simplicity of a fixed coarsening scheme is beneficial from both a conceptual and software perspective. Fixed coarsening schemes are more amenable to a Graphical Processing Unit (GPU) implementation

and more efficient in terms of memory access patterns. Fixed coarsening schemes are also a good match for smoothly varying problems. However, when the weights are highly irregular (jumping coefficients), such a strategy is restrictive and can slow down the convergence of the overall system. An example of such a problem comes from thin, long edges that span across a large part of the image. Pixels on either side of the edge are poorly connected with pixels along the edge. A fixed red-black coarsening strategy interpolates the coarse level solution from the pixels around the edge to pixels on the edge, and this slows down the overall convergence to the solution. The interpolated solutions from the coarse levels are no longer a close approximation to the fine level solutions, which a good interpolation operator should achieve. For such highly irregular problems, adaptive coarsening strategies such as those in (Koutis, Miller, and Tolliver 2009; Trottenberg, Oosterlee, and Schuller 2000) are required to achieve good convergence.

8 Discussion and Conclusions

Our experimental results demonstrate that with our new extension to partial sparsification, adaptive hierarchical basis function preconditioning (ABF) outperforms traditional multigrid techniques on a wide range of computer graphics and computational photography problems. This runs contrary to the usual belief in the multigrid community that appropriate smoothing between level transfers is necessary for reasonable performance. Traditional adaptive hierarchical basis functions achieve slower (but still reasonable) convergence rates without using any smoothing steps, but use many fewer computations per level because they do not require any smoothing (other than that implicit in the recomputation of the residual at the beginning of each iteration).

For irregular problems, such as edge-preserving decomposition, adaptive coarsening schemes such as combinatorial and (adaptive) algebraic multigrid start to become important. Our experiments also show that taking into account only the number of iterations without incorporating flop counts gives a misleading picture of relative performance. However, we note that flop counts are themselves not a complete predictor of actual performance, since issues such as caching and memory access patterns will be important in high-performance implementations of these algorithms.

Recently, a number of fast edge-aware smoothing techniques have been developed for solving tone mapping and colorization problems (Fattal 2009; Gastal and Oliveira 2011). These techniques have linear complexity in the number of pixels and have extremely fast computational performance. However, the best-performing preconditioning methods in our paper also converge in a single iteration for tone mapping and colorization. Therefore the effective time for processing is the multilevel pyramid setup time and a

single iteration of (preconditioned) conjugate gradient. Both these steps are a small constant multiple of the number of pixels in the image and are highly parallel operations. This leads us to believe that a carefully optimized GPU version of our MATLAB code would give very fast computational performance as well for the tone mapping and colorization algorithms.

For the other algorithms such as Poisson blending and edge-preserving decomposition, solving a large linear system is a necessity. Our MATLAB code has been written explicitly with GPU implementation in mind, since such an implementation would be of significant practical interest to the graphics and computer vision communities.

Currently, our adaptive hierarchical bases are defined on a fixed half-octave grid. However, we believe that the coarsening heuristics used in AMG could be adapted to our bases. The sparsification step used in constructing the adaptive interpolants requires a decomposition into coarse and fine variables, where the connections between fine variables (which need to be sparsified) are no stronger than those to their corresponding coarse-level "parents". AMG coarsening strategies perform a very similar selection of nodes.

Another promising direction for future research is the extension to three-dimensional and more unstructured meshes. The advances available by combining all of these ideas together should lead to even more efficient techniques for the solution of large-scale two- and three-dimensional computer graphics and computational photography problems.

A Jacobi smoothing

In this section, we derive explicit formulae for the iteration matrices corresponding to Jacobi smoothing. We then use these formulae in Appendix B to derive an explicit expression for a 2-level multigrid preconditioner with Jacobi smoothing.

Consider ω-damped Jacobi smoothing to solve the problem $Ax = b$ starting at the initial iterate x_0 (Trottenberg, Oosterlee, and Schuller 2000), when A is a symmetric, diagonally dominant, and positive definite matrix. Here $0 < \omega < 1$. Let the diagonal of A be given by D. Given iterate x_{k-1} at the $k-1^{\text{st}}$ iteration, the iterate x_k at the next iteration is given by

$$\begin{aligned} x_k &= x_{k-1} + \omega D^{-1}(b - Ax_{k-1}) && (27)\\ &= \omega D^{-1}b + (I - \omega D^{-1}A)x_{k-1} && (28)\\ &= \omega D^{-1}b + Hx_{k-1}, && (29) \end{aligned}$$

where

$$H = I - \omega D^{-1} A \tag{30}$$

is called the *iteration matrix*[10] and $I - H = \omega D^{-1} A$. Expanding x_{k-1} in terms of x_{k-2} gives

$$\begin{aligned} x_k &= \omega D^{-1} b + H(\omega D^{-1} b + H x_{k-2}) && (31) \\ &= \omega(I + H) D^{-1} b + H^2 x_{k-2} && (32) \end{aligned}$$

Hence, after k iterations, we get

$$\begin{aligned} x_k &= \omega[I + H + \ldots + H^{k-1}] D^{-1} b + H^k x_0 && (33) \\ &= \omega(I - H^k)(I - H)^{-1} D^{-1} b + H^k x_0 && (34) \\ &= (I - H^k) A^{-1} b + H^k x_0 && (35) \\ &= x^* + H^k (x_0 - x^*), && (36) \end{aligned}$$

where $x^* = A^{-1} b$ is the optimal (final) solution. For convenience, we can write

$$R_k = \omega \sum_{i=0}^{k-1} H^i D^{-1} = \omega(I - H^k)(I - H)^{-1} D^{-1} = (I - H^k) A^{-1} \tag{37}$$

(Tatebe 1993) and

$$x_k = R_k b + H^k x_0. \tag{38}$$

Since A is SDD, the entries of $H = (I - \omega D^{-1} A)$ are less than 1 in absolute value. As k becomes larger, all entries in H^k approach 0. If we just have $k = 1$ (one iteration), Jacobi smoothing simplifies to

$$x_1 = \omega D^{-1} b + H x_0 \tag{39}$$

B 2-Level multigrid preconditioner

Multigrid methods that satisfy certain conditions may be used in conjunction with CG as a preconditioner. In this section, we derive an explicit matrix form for a 2-level MG preconditioner that uses Jacobi smoothing. Within a CG iteration, the preconditioner is applied to the residual at every iteration. To derive a 2-level preconditioner, we make a slight change of notation from the previous section. We use the subscripts l and $l + 1$ to denote fine and coarse-level variables and use the superscript k (where needed for clarity) to denote the iteration count. The fine-level system that we seek to solve is then given by $A_l x_l = b_l$.

[10] The iteration matrix is denoted by M in (Trottenberg, Oosterlee, and Schuller 2000), R in (Briggs, Henson, and McCormick 2000), G in (Saad 2003), and H in (Tatebe 1993).

Let the current iterate at the fine level be given by x_l^k. Let the fine-level matrix be given by A_l. Define the residual at the fine level as $r_l = b_l - A_l x_l^k$. If the optimal solution is x_l^*, the error at the fine level is given by $e_l = x_l^* - x_l^k$. The fine-level equation that links the error e_l to the residual r_l is therefore $A_l e_l = r_l$. This is easily derived from the original equation $A_l x_l = b_l$.

Any multigrid or multilevel method we use is associated with a particular interpolation matrix that generates fine-level variables from coarse-level variables. Denote this interpolation matrix as $\hat{S}_l$ and the corresponding restriction matrix as $\hat{S}_l^T$. $\hat{S}_l$ is rectangular and contains more rows than columns (since there are more fine-level variables than coarse). Let A_{l+1} be the coarse-level Hessian matrix. By the Galerkin condition (Trottenberg, Oosterlee, and Schuller 2000), $A_{l+1} = \hat{S}_l^T A_l \hat{S}_l$. If A_l is symmetric positive definite, so is A_{l+1} if $\hat{S}_l$ is of full column rank. However, the diagonal dominance of A_{l+1} is not assured for all $\hat{S}_l$.

The equation we are trying to approximately solve at each PCG step is $A_l e_l = r_l$. If we had a perfect preconditioner, it would simply invert A_l and the resulting preconditioned r_l would be $A_l^{-1} r_l$. Adding this to the current iterate would then result in a single-step solution to the problem. To find the perfect preconditioner, however, is prohibitively expensive. Therefore, our task is to find a good approximation to the true error.

To precondition the residual r_l, multigrid uses the following steps: pre-smoothing, residual restriction, coarse-level solution, error interpolation, and post-smoothing. It is well-known (Tatebe 1993) that to generate a symmetric positive definite preconditioner, the number of pre-smoothing steps must be equal to that of post-smoothing, and this number must be greater than 0 for multigrid methods. Hence, let us consider $k > 0$ Jacobi smoothing steps at the pre-smoothing and post-smoothing stages. Based on Appendix A, let us re-define Eqn. 30 and Eqn. 37 as follows:

$$H_l = I - \omega D_l^{-1} A_l \tag{40}$$

$$R_l = \omega \sum_{i=0}^{k-1} H_l^i D_l^{-1} \tag{41}$$

The task of pre-smoothing and post-smoothing is complementary to that of the coarse-level operations: smoothing quickly finds the oscillatory components of the true error and the coarse-level operations find the smooth components of the true error. We therefore start with an assumption that the true error is 0. The pre-smoothing finds some oscillatory components of the error. We form the residual corresponding to this error and use coarse-level operations to find the smoother components of the true error. The reason to switch to a coarser grid is that smooth errors on a fine grid appear more oscillatory on a coarser grid and can therefore be reduced by Jacobi iterations at the coarser level (Briggs, Henson, and McCormick 2000). We then add these smoother components back to the error found after the initial pre-smoothing. Finally, to

reduce any oscillatory components of error that have been introduced due to the coarsening and interpolation operations, we perform post-smoothing. These steps are now given in detail below, and will progressively form the components of the preconditioner matrix.

1. *Pre-smoothing*: The task in pre-smoothing is to find an approximation to the oscillatory components of the true error. We run k iterations of Jacobi with $e_l^0 = 0$. This gives (Eqn. 38):

$$e_l^1 = R_l r_l \tag{42}$$

2. *Residual Computation*: The next task is to find the remaining error components that have not been reduced by the Jacobi smoothing. These are usually the smooth components. To do this, we have to solve a residual equation, where the residual is computed by removing the components in e_l^1. Hence, we compute

$$\tilde{r}_l = r_l - A_l e_l^1 = (I_l - A_l R_l) r_l \tag{43}$$

3. *Restriction*: To remove smooth components faster, we now move to a coarser level and solve the new residual problem there. The restriction of the residual to the coarse-level is given by

$$\tilde{r}_{l+1} = \hat{S}_l^T r_l = \hat{S}_l^T (I_l - A_l R_l) r_l \tag{44}$$

4. *Coarse-Level Solve*: At the coarse level, we may either have an approximate or exact solution, depending on the size of the coarsened problem. Let $\bar{A}_{l+1}$ be an approximation to the coarse-level Hessian. If we want an exact solution at the coarse level, then $\bar{A}_{l+1} = A_{l+1}$. In a multi-level (more than 2 level) preconditioner, this step is recursively replaced by the same smooth-coarsen-smooth preconditioner we are now analyzing. In that case, we have an approximate inverse. For a V-cycle based preconditioner, the coarse level problem is solved once to give a coarse error vector $\tilde{e}_{l+1}$:

$$\begin{aligned} \tilde{e}_{l+1} &= \hat{A}_{l+1}^{-1} \tilde{r}_{l+1} && (45) \\ &= \hat{A}_{l+1}^{-1} \hat{S}_l^T (I_l - A_l R_l) r_l && (46) \end{aligned}$$

   If instead, we have a W-cycle based preconditioner, then the coarse level problem is repeatedly solved for ν cycles with update to the coarsened error vector. This gives rise to the following formula:

$$\begin{aligned} \tilde{e}_{l+1} &= \hat{A}_{l+1}^{-1} \Big(\sum_{k=0}^{\nu-1} (I - A_{l+1} \hat{A}_{l+1}^{-1})^k\Big) \tilde{r}_{l+1} && (47) \\ &= \hat{A}_{l+1}^{-1} \Big(\sum_{k=0}^{\nu-1} (I - A_{l+1} \hat{A}_{l+1}^{-1})^k\Big) \hat{S}_l^T (I_l - A_l R_l) r_l && (48) \\ &= \tilde{A}_{l+1} \hat{S}_l^T (I_l - A_l R_l) r_l && (49) \end{aligned}$$

where we have defined

$$\tilde{A}_{l+1} = \hat{A}_{l+1}^{-1}(\sum_{k=0}^{\nu-1}(I - A_{l+1}\hat{A}_{l+1}^{-1})^k) \tag{50}$$

5. *Prolongation*: We now transfer the error computed at the coarse level back to the fine level:

$$\tilde{e}_l = \hat{S}_l\tilde{e}_{l+1} = \hat{S}_l\tilde{A}_{l+1}^{-1}\hat{S}_l^T(I_l - A_lR_l)r_l \tag{51}$$

6. *Error Correction*: The error components determined from the coarse-level computations must be added to the error components determined from the pre-smoothing Jacobi iterations:

$$e_l = e_l^1 + \tilde{e}_l = [R_l + \hat{S}_lA_{l+1}^{-1}\hat{S}_l^T(I_l - A_lR_l)]r_l \tag{52}$$

7. *Post-smoothing*: Since the interpolation and restriction operators have re-introduced oscillatory components into the error, we perform post-smoothing. This uses the current iterate e_l and the right hand side r_l to give the final corrected error e_l^*. Using the formula for e_l above and the Jacobi smoothing formula Eqn. 38, we get:

$$\begin{aligned} \tilde{r}_l &= R_lr_l + H_l^ke_l & (53)\\ &= [R_l + H_l^kR_l + H_l^k\hat{S}_l\tilde{A}_{l+1}^{-1}\hat{S}_l^T(I_l - A_lR_l)]r_l & (54)\end{aligned}$$

The action of the 2-level multigrid preconditioner on the residual r_l may therefore be written as (using Eqns. 40–41):

$$\tilde{r}_l = M^{MG}r_l. \tag{55}$$

where

$$M^{MG} = R_l + H_l^kR_l + H_l^k\hat{S}_l\tilde{A}_{l+1}^{-1}\hat{S}_l^T(I_l - A_lR_l). \tag{56}$$

The preconditioner above is symmetric and positive definite. The symmetry is seen as follows, using Eqns. 40–41:

$$\begin{aligned} (H_lD_l^{-1})^T = D_l^{-1}(I_l - \omega A_lD_l^{-1}) = H_lD_l^{-1} & & (57)\\ \Rightarrow (H_l^2D_l^{-1})^T = D_l^{-1}H_l^TH_l^T = H_l^2D_l^{-1} & & (58)\\ \Rightarrow (H_l^iD_l^{-1})^T = H^iD_l^{-1} \Rightarrow R_l^T = R_l, (H_l^kR_l)^T = (H_l^kR_l) & & (59)\end{aligned}$$

$$(I_l - A_l R_l)^T = I_l - R_l^T A_l \tag{60}$$

$$= I_l - R_l^T (\omega D_l^{-1})^{-1} (I_l - H_l) \tag{61}$$

$$= I_l - \sum_{i=0}^{k-1} (H_l^i (\omega D_l^{-1}))(\omega D_l^{-1})^{-1} (I_l - H_l) \tag{62}$$

$$= I_l - \sum_{i=0}^{k-1} (H_l^i - H_l^{i+1}) = H_l^k. \tag{63}$$

Now, for symmetric diagonally dominant matrices A_l, the matrix H_l has eigenvalues with absolute value less than 1 (Saad 2003). In Eqn. 56, the third term is of the form $H_l^k \hat{S}_l A_{l+1}^{-1} \hat{S}_l^T (H_l^k)^T$, which is symmetric positive *semi-definite* if $\tilde{A}_{l+1}$ is symmetric positive definite. Let us consider the sum of the first 2 terms: $R_l + H_l^k R_l$. This can be written as $V = \omega \sum_{i=0}^{2k-1} H_l^i D_l^{-1}$. The eigenvalues of V^T are the same as those of V, and $V^T = \omega \sum_{i=0}^{2k-1} D_l^{-1} (H_l^i)^T$. If λ is an eigenvalue of H_l^T with corresponding eigenvector v, then $D_l^{-1} H_l^T v = \lambda D_l^{-1} v$. Hence, $\omega D_l^{-1} (H_l^i)^T v = \omega \lambda^i D_l^{-1} v$. Hence, we see that the eigenvalues of V are given by $\sum_{i=0}^{2k-1} \omega \lambda^i = \omega \frac{\lambda^{2k}-1}{\lambda - 1}$, which is positive if $|\lambda| < 1$. Hence V is positive definite and therefore, the multigrid preconditioner in Eqn. 56 is symmetric positive definite. Finally, the symmetry and positive definiteness of $\tilde{A}_{l+1}$ is guaranteed by the construction of the coarse level matrices.

C Gauss-Seidel smoothing and 2-Level multigrid preconditioner

Raster-order Gauss-Seidel (GS) smoothing is a classical iterative smoother. In Jacobi smoothing, updates are entirely parallel in nature since every point is updated using values from the previous iterate. In contrast, raster-order or lexicographic GS (GS-LEX in (Trottenberg, Oosterlee, and Schuller 2000)), updates points in sequential order. The latest value of each point is used to points which arrive later in raster order. The loss of parallelism in comparison to Jacobi smoothing is offset by an increased convergence rate.

Gauss-Seidel smoothing has a number of variants. These involve splitting the points into disjoint groups, such that points within each group only have neighbors that in other groups. Points in one group may then be updated together, possibly in parallel. When updating a group, the latest values from the other groups are used (otherwise we just revert to Jacobi smoothing). When the groups correspond to red-black ordering, this variant is known as GS-RB. GS-RB is completely parallel for 5-point stencils, but not so for 9-point stencils. When the points are split into 4 groups such that points within each group only have neighbors from the other 3 groups, this is known as four-color pointwise GS (GS-FC). See (Trottenberg, Oosterlee, and Schuller 2000) for details.

Consider GS-LEX to solve the problem $Ax = b$ starting at the initial iterate x_0, where A arises from

either a 5-point or 9-point stencil. Let $A = D + E + F$, where D is the main diagonal of A, E is the strictly lower triangular part of A and L is the strictly upper triangular part. Since A is symmetric, $E^T = F$. Given iterate x^{k-1}, the iterate x_k is given by

$$\begin{aligned}
(D+E)x_k &= b - Fx_{k-1} \\
\Rightarrow x_k &= (D+E)^{-1}b - (D+E)^{-1}Fx_{k-1} \\
&= (D+E)^{-1}b - (D+E)^{-1}F((D+E)^{-1}b - (D+E)^{-1}Fx_{k-2}) \\
&= [I - (D+E)^{-1}F + \ldots](D+E)^{-1}b + (-1)^k((D+E)^{-1}F)^k x_0 \\
&= Rb + H^k x_0 \qquad (64)
\end{aligned}$$

where,

$$H = -(D+E)^{-1}F \qquad (65)$$

$$R = [\Sigma_{m=0}^{k-1} H^m](D+E)^{-1} = [I - H^k](I-H)^{-1}(D+E)^{-1} = [I - H^k]A^{-1} \qquad (66)$$

Following the same derivation as above, we can see that doing GS updates in reverse raster order requires swapping the roles of E and F and gives rise to the following matrices

$$\tilde{H} = -(D+F)^{-1}E \qquad (67)$$

$$\tilde{R} = [\Sigma_{m=0}^{k-1} \tilde{H}^m](D+F)^{-1} = [I - \tilde{H}^k](I-\tilde{H})^{-1}(D+F)^{-1} = [I - \tilde{H}^k]A^{-1} \qquad (68)$$

For a matrix A, we call the associated matrices $R, H, \tilde{R}, \tilde{H}$ the iteration matrices. Next, we consider the construction of a 2-level multigrid preconditioner based on GS-LEX smoothing. To ensure the symmetry and positive-definiteness of the preconditioner, we assume one pre-smoothing and post-smoothing step. Furthermore, the raster-order in the pre-smoothing and post-smoothing must be the opposite of each other to maintain symmetry and positive definiteness, (Tatebe 1993). Following the same terminology as in Appendix B, let A_l represent the Hessian matrix at level l, with $A_l = D_l + E_l + F_l$; and let the iteration matrices be represented as $R_l, H_l, \tilde{R}_l, \tilde{H}_l$. As before, let $\hat{S}_l$ represent the interpolation matrix, and A_{l+1} the coarse level Hessian matrix, which we assume is SPD. The preconditioner associated with GS-LEX may be written as

$$M_{GS-LEX}^{MG} = \tilde{R}_l + \tilde{H}_l R_l + \tilde{H}_l \hat{S}_l A_{l+1}^{-1} \hat{S}_l^T (I_l - A_l R_l). \qquad (69)$$

The symmetry of M^{MG}_{GS-LEX} may be seen by using the fact that $E^T = F$ and that D is diagonal. Then

$$\begin{aligned}(I - A_l R_l)^T &= I - R_l^T A_l^T \\ &= I - (D_l + E_l)^{-T}(D_l + E_l^T + F_l^T) \\ &= -(D_l + F_l)^{-1} E_l \\ &= \tilde{H}_l\end{aligned}$$

For the first 2 terms in Eqn. 69,

$$\begin{aligned}\tilde{R}_l + \tilde{H}_l R_l &= \\ &= (D_l + F_l)^{-1} - (D_l + F_l)^{-1} E (D_l + E_l)^{-1} \\ &= (D_l + F_l)^{-1}(I - E_l(D_l + E_l)^{-1}) \\ &= (D_l + F_l)^{-1}(D_l + E_l - E_l)(D_l + E_l)^{-1} \\ &= (D_l + F_l)^{-1} D_l (D_l + E_l)^{-1}\end{aligned}$$

which is symmetric, and since the elements of D are greater than 0 it is also positive definite. Hence M^{MG}_{GS-LEX} is symmetric. With regards to positive definiteness, it is clear that due to A_{l+1} being SPD, the last term $\tilde{H}_l \hat{S}_l A_{l+1}^{-1} \hat{S}_l^T (I_l - A_l R_l)$ is symmetric positive semi-definite. Putting all this together shows that the preconditioner is SPD.

For GS variants such as GS-RB and GS-FC, points in one group are updated first, then points in the next group are updated using the latest values in the already-updated groups. This is equivalent to permuting the raster-order to follow the group order, applying GS-LEX, and finally re-arranging back to raster order. Let a permutation matrix P be defined, such that $\bar{x} = Px$ re-arranges points in x according to the group order. P^T does the reverse, i.e. takes points in group order and re-arranges them to raster-order. Since P is a permutation matrix, it is orthogonal. Hence $Ax = b$ is equivalent to $PAP^T Px = Pb$ which can be re-written as $\bar{A}\bar{x} = \bar{b}$ with $\bar{A} = PAP^T$. From Eqn. 64 $\bar{x}_k = \bar{R}\bar{b} + \bar{H}^k \bar{x}_0$, with $\bar{R}$ and $\bar{H}$ being the matrices analogous to Eqn. 66 and Eqn. 65 for $\bar{A}$. Finally, re-arranging the points gives $x_k = P^T \bar{x}_k = P^T \bar{R} P b + P^T \bar{H}^k P x_0$. The iteration matrices for GS-RB and GS-FC are therefore of the same form as that of GS-LEX, with the addition of the permutation matrices. So are the corresponding preconditioning matrices. The preconditioner analysis remains the same with the addition of permutation matrices.

D Unified presentation of multilevel preconditioners and multigrid methods

This section gives a unified description of multilevel preconditioners, with and without the incorporation of smoothing techniques. Let us assume that we are given a general smoothing method of the form, k iterations of which results in an iterate $x_k = Rb + H^k x_0$, where x_0 is the initial iterate. If no smoothing is used, then $R = 0, H = I$. Let the Hessian at the fine level be given by A_l, and that at the coarse level be given by A_{l+1}. We assume that both of these matrices are symmetric and positive definite. Furthermore, the prolongation operator that takes variables from the coarse level to the fine level is given by S_l, and the restriction operator that reverses this is S_l^T. Let the operators for pre-smoothing and post-smoothing be given by R_l^{pre}, H_l^{pre} and R_l^{post}, H_l^{post} respectively.

Let the current iterate be given by x_k. The goal of a multilevel or multigrid technique is to generate the next iterate x_{k+1} that gives accelerated convergence towards the true solution $x = A_l^{-1} b$. Our aim is to find a correction e_k such that $x_{k+1} = x_k + \alpha_k e_k$, where α_k is an appropriate step size. For multilevel methods used as a preconditioner with conjugate gradients, α_k is determined by an appropriate step size computation (see Algorithm 3). For V cycles or W cycles, α_k is always 1.

The residual r_k at the current iteration is given by $r_k = b - A_l x_k$. The steps applied in a multilevel preconditioners or V-sweep are: optional pre-smoothing, coarse level solve, and optional post-smoothing. We start with error $e_k = 0$. If there is no pre-smoothing, then $e_k^{pre} = 0$, otherwise $e_k^{pre} = R_l r_k$. Pre-smoothing (if used) determines the higher frequency components of the error in a small number of iterations. The lower frequency components are very slowly recovered by most smoothers. To overcome this, we need to use coarse-grid correction. First, we compute a new residual $\bar{r}_k = r_k - A_l e_k^{pre}$. Then this residual is restricted to the coarse level using the restriction operator $\tilde{r}_k = S_l^T \bar{r}_k$.

The coarse level may either be solved directly or approximately. If the coarse level solve is exact, we compute $\tilde{e}_k = A_{l+1}^{-1} \tilde{r}_k$. If not, we compute $\tilde{e}_k = \hat{A}_{l+1}^{-1} \tilde{r}_k$, where $\hat{A}_{l+1}$ is an approximation to A_{l+1}. Let us use M_{l+1} to denote the either A_{l+1}^{-1} or $\hat{A}_{l+1}^{-1}$ depending on whether we use an exact solve or an approximate solve. Hence, $\tilde{e}_k = M_{l+1} \tilde{r}_k$. If more than one coarse level solve is used then the following sequence of

operations is done, where I_{l+1} denotes the identity matrix at the coarse level:

$$\begin{aligned}
\tilde{e}_k &= M_{l+1}\tilde{r}_k && (70)\\
\tilde{r}_k &\equiv \tilde{r}_k - A_{l+1}\tilde{e}_k = (I_c - A_{l+1}M_{l+1})\tilde{r}_k && (71)\\
\tilde{e}_k &= M_c\tilde{r}_k = (I_{l+1} - M_{l+1}A_{l+1})M\tilde{r}_k && (72)\\
\tilde{r}_k &\equiv \tilde{r}_k - A_{l+1}\tilde{e}_k = [I_{l+1} - A_{l+1}(M_{l+1} + M_{l+1}(I_{l+1} - A_{l+1}M_{l+1}))]\tilde{r}_k && (73)\\
\tilde{e}_k &= M_{l+1}\tilde{r}_k = (I_{l+1} - M_{l+1}A_{l+1})^2M\tilde{r}_k && (74)\\
& && (75)
\end{aligned}$$

This is repeated for as many cycles as desired. Hence one update is done for a V-cycle based preconditioners and two updates for W-cycle preconditioners. It is seen from the above that the formula for $\tilde{e}_k$ may be expressed as follows, where k is the number of updates ($k = 1$ for V-cycles, $k = 2$ for W-cycles)

$$\tilde{e}^k = \sum_{j=0}^{k-1}(I_{l+1} - M_{l+1}A_{l+1})^jM\tilde{r}_k = [I_{l+1} - (I_{l+1} - M_{l+1}A_{l+1})^k]A_{l+1}^{-1}\tilde{r}_k \quad (76)$$

Next the coarse level correction $\tilde{e}^k$ is interpolated to the fine-level. This gives the coarse error correction at the fine level:

$$\bar{e}_k = S_l\tilde{e}_k \quad (77)$$

In an multilevel preconditioner framework such as ABF and HBF, an additional step of diagonal preconditioning is done for the fine-level variables only. This is performed on the residual $\tilde{r}_k$. Let $\tilde{D}_l$ denote a diagonal matrix whose entries are the diagonal entries of A_l, but with zeros in coarse variables. In pure multigrid methods, let $\tilde{D}_l$ be zeros at all variables. Then the optional diagonal preconditioning modified $\bar{e}_k$ as

$$\bar{e}_k \equiv \bar{e}_k + \tilde{D}_l\tilde{r}_k \quad (78)$$

The error components from pre-smoothing and coarse level correction are added to give an error $e_k = \bar{e}_k + e_k^{Pre}$. If there is post-smoothing, then e_k is modified using the residual r_k as

$$e_k^{Post} = R_l^{Post}r + H_l^{Post}e_k \quad (79)$$

Expanding all the above steps gives us that $e_k^{Post} = M_lr_k$, where

$$M_l = R_l^{Post} + H_l^{Post}[R_l^{Pre} + (\tilde{D}_l + S_l(I_{l+1} - (I_{l+1} - M_{l+1}A_{l+1})^k)A_{l+1}^{-1}S_l^T))(I_l - A_lR_l^{Pre}] \quad (80)$$

The matrix M_l is then an approximation to A_l^{-1}. It can be shown by induction that the matrix $I_{l+1} - (I_{l+1} - M_{l+1}A_{l+1})^k)A_{l+1}^{-1}$ is symmetric for any k if M_{l+1} is symmetric. Then it can be shown that for any smoother

such as red-black Gauss-Seidel M_l is symmetric and positive definite. If no smoothing is used, then M_l is symmetric and positive definite if $\tilde{D}_l$ consists of positive values at the fine-level variables.

E Full set of experimental results

This Appendix contains a summary of all the experiments performed in writing this paper.

For every problem and data set size, we present the convergence rate results in tabular form and as plots of error vs. both iteration count and computational cost (100s of flops). For all but the largest problems, we also provide theoretical convergence rates based on the generalized condition numbers of the preconditioners (for preconditioned conjugate gradient) or the spectral radius of the iteration matrix (Section 4).

In order to distinguish these complete results from tables and figures earlier in the paper and to make it easier to associate corresponding table and figure numbers, we re-start the table and figure numbering at 101.

For each algorithm and problem, we use multiple levels of preconditioning and report the convergence results. The following quantities are reported in the theoretical convergence tables:

1. κ: The preconditioned condition number. For problems of moderate size, we use power iterations to determine the smallest and largest eigenvalues of the preconditioned system. In some cases, these power iterations did not converge; hence we do not report κ for those cases and instead place a blank there. In addition for larger problems, the W-cycles take a very long time for power iterations and we do not report κ for larger problem W-cycles.

2. κ_0: The condition number of the coarsest level Hessian.

3. ρ: Theoretical asymptotic convergence rate Eqn. 17.

4. τ: Negative natural log of ρ.

5. C: Computation load (flops) per fine level point per iteration.

6. $\hat{\tau}$: Effective theoretical convergence rate $\hat{\tau} = 100\,\tau/C$.

7. The original condition number of the system is reported below the last row of the table.

The following quantities are reported in the empirical convergence tables:

1. $\tilde{\rho}$: Empirical asymptotic convergence rate based on how much the error with respect to ground-truth is reduced.

2. $\tilde{\tau}$: Negative natural log of $\tilde{\rho}$.

3. C: Computation load (flops) per fine level point per iteration.

4. $\hat{\tau}$: Effective empirical convergence rate $\hat{\tau} = 100\,\tilde{\tau}/C$.

5. The original condition number of the system is reported below the last row of the table.

References

Agarwala, A. (2007). Efficient gradient-domain compositing using quadtrees. *ACM Transactions on Graphics (Proc. SIGGRAPH 2007)*, 26(3):94:1–94:5.

Agarwala, A. *et al*. (2004). Interactive digital photomontage. *ACM Transactions on Graphics (Proc. SIGGRAPH 2004)*, 23(3):292–300.

Alon, N. and Yuster, R. (2010). Solving linear systems through nested dissection. *Proc. of FOCS 2010*.

Briggs, W. L. (1987). *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, Philadelphia.

Briggs, W. L., Henson, V. E., and McCormick, S. F. (2000). *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, Philadelphia, second edition.

Davis, T. A. (2006). *Direct Methods for Sparse Linear Systems*. SIAM.

Farbman, Z., Fattal, R., Lischinski, D., and Szeliski, R. (2008). Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics (Proc. SIGGRAPH 2008)*, 27(3).

Farbman, Z., Hoffer, G., Lipman, Y., Cohen-Or, D., and Lischinski, D. (2009). Coordinates for instant image cloning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2009)*, 28(3).

Fattal, R. (2009). Edge-avoiding wavelets and their applications. *ACM Transactions on Graphics (Proc. SIGGRAPH 2009)*, 28(3).

Fattal, R., Lischinski, D., and Werman, M. (2002). Gradient domain high dynamic range compression. *ACM Transactions on Graphics*, 21(3):249–256.

Gastal, E. S. L. and Oliveira, M. M. (2011). Domain transform for edge-aware image and video processing. *ACM Transactions on Graphics (Proc. SIGGRAPH 2011)*, 30(4).

Golub, G. and Van Loan, C. F. (1996). *Matrix Computation, third edition*. The John Hopkins University Press, Baltimore and London.

Gortler, S. J. and Cohen, M. F. (1995). Hierarchical and variational geometric modeling with wavelets. In *Symposium on Interactive 3D Graphics*, pp. 35–43, Monterey, CA.

Grady, L. (2006). Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783.

Grady, L. (2008). A lattice-preserving multigrid method for solving the inhomogeneous Poisson equations used in image analysis. In , pp. 252–264, Marseilles.

Jeschke, S., Cline, D., and Wonka, P. (2009). A GPU Laplacian solver for diffusion curves and Poisson image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2009)*, 28(5).

Kazhdan, M., Surendran, D., and Hoppe, H. (2010). Distributed gradient-domain processing of planar and spherical images. *ACM Transactions on Graphics*, 29(2).

Koutis, I. and Miller, G. L. (2008). Graph partitioning into isolated, high conductance clusters: theory, computation and applications to preconditioning. In *Symposium on Parallel Algorithms and Architectures*, pp. 137–145, Munich.

Koutis, I., Miller, G. L., and Tolliver, D. (2009). Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing. In *5th International Symposium on Visual Computing (ISVC09)*, Las Vegas.

Kushnir, D., Galun, M., and Brandt, A. (2010). Efficient multilevel eigensolvers with applications to data analysis tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1377–1391.

Lai, S.-H. and Vemuri, B. C. (1997). Physically based adaptive preconditioning for early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):594–607.

Levin, A., Lischinski, D., and Weiss, Y. (2004). Colorization using optimization. *ACM Transactions on Graphics*, 23(3):689–694.

Levin, A., Lischinski, D., and Weiss, Y. (2008). A closed-form solution to natural image matting. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30(2):228–242.

Levin, A., Zomet, A., Peleg, S., and Weiss, Y. (2004). Seamless image stitching in the gradient domain. In *Eighth European Conference on Computer Vision (ECCV 2004)*, pp. 377–389, Prague.

Lischinski, D., Farbman, Z., Uyttendaele, M., and Szeliski, R. (2006). Interactive local adjustment of tonal values. *ACM Transactions on Graphics (Proc. SIGGRAPH 2006)*, 25(3):646–653.

McAdams, A., Sifakis, E., and Teran, J. (2010). A parallel multigrid Poisson solver for fluids simulation on large grids. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, :65–74.

McCann, J. and Pollard, N. (2008). Real-time gradient domain painting. *ACM Transactions on Graphics (Proc. SIGGRAPH 2008)*, 27(3).

Napov, A. and Notay, Y. (2011). Algebraic analysis of aggregation-based multigrid. *Numerical Linear Algebra with Applications*, 18(3):539–564.

Notay, Y. (2010). An aggregation-based algebraic multigrid method. *Electronic Transactions on Numerical Analysis*, 37:123–146.

Notay, Y. and Amar, Z. O. (1997). A nearly optimal preconditioning based on recursive red-black ordering. *Numerical Linear Algebra with Applications*, 4:369–391.

Pentland, A. P. (1994). Interpolation using wavelet bases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(4):410–414.

Pérez, P., Gangnet, M., and Blake, A. (2003). Poisson image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, 22(3):313–318.

Roberts, A. (2001). Simple and fast multigrid solutions of Poisson's equation using diagonally oriented grids. *ANZIAM J. 43*, :E1–E36.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition.

Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain. Unpublished manuscript, available on author's homepage (`http://www.cs.berkeley.edu/~jrs/`). An earlier version appeared as a Carnegie Mellon University Technical Report, CMU-CS-94-125.

Szeliski, R. (1990). Fast surface interpolation using hierarchical basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):513–528.

Szeliski, R. (1991). Fast shape from shading. *CVGIP: Image Understanding*, 53(2):129–153.

Szeliski, R. (2006). Locally adapted hierarchical basis preconditioning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2006)*, 25(3):1135–1143.

Szeliski, R., Uyttendaele, M., and Steedly, D. (2011). Fast Poisson blending using multi-splines. In *International Conference on Computational Photography (ICCP 11)*, Pittsburgh.

Tatebe, O. (1993). The multigrid preconditioned conjugate gradient method. In *Sixth Copper Mountain Conference on Multigrid Methods*, pp. 621–634, Copper Mountain, Colorado.

Terzopoulos, D. (1983). Multilevel computational processes for visual surface reconstruction. *Computer Vision, Graphics, and Image Processing*, 24:52–96.

Terzopoulos, D. (1986). Image analysis using multigrid relaxation methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(2):129–139.

Trottenberg, U., Oosterlee, C. W., and Schuller, A. (2000). *Multigrid*. Academic Press.

Wang, H., Raskar, R., and Ahuja, N. (2004). Seamless video editing. *Proceedings of the International Conference on Pattern Recognition*, :858–861.

Yaou, M.-H. and Chang, W.-T. (1994). Fast surface interpolation using multiresolution wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(7):673–689.

Complete Experimental Results for
Comparison of Multigrid and Multilevel Preconditioners

	$L=1$				$L=2$				$L=3$				$L=4$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.19	1.65	43.7	3.78	0.20	1.62	35.6	4.55	0.28	1.27	34.6	3.67	0.31	1.16	34.6	3.35
ABF-Pre7	0.02	3.80	49.7	7.64	0.05	3.09	43.3	7.13	0.08	2.47	42.8	5.76	0.10	2.33	42.9	5.42
ABF-Pre7-4C	0.00	9.79	79.2	12.36	0.00	8.51	96.6	8.81	0.00	7.69	102.8	7.48	0.00	7.51	105.0	7.15
AMG-J	0.02	4.11	80.7	5.09	0.02	4.07	88.9	4.58	0.02	4.08	92.4	4.41	0.02	4.10	93.8	4.37
AMG-4C	0.00	6.71	76.5	8.77	0.00	6.74	83.6	8.06	0.00	6.63	86.9	7.63	0.00	6.64	88.2	7.53
GMG-J	0.01	4.64	83.2	5.57	0.01	4.61	92.1	5.00	0.01	4.57	95.9	4.76	0.01	4.53	97.4	4.65
V(1,1)-4C	0.00	5.92	72.2	8.20	0.00	5.85	79.3	7.38	0.00	5.81	82.6	7.04	0.00	5.82	83.8	6.95
V(0,1)-4C	0.05	3.05	49.2	6.21	0.07	2.72	45.9	5.91	0.10	2.32	46.3	5.00	0.11	2.21	46.7	4.74
CMG	0.28	1.26	60.6	2.08	0.29	1.24	69.1	1.80	0.33	1.09	73.4	1.49	0.34	1.09	80.5	1.36
ABF-Pre7-W	0.00	7.45	79.4	9.39	0.00	7.56	119.4	6.33	0.00	7.56	170.2	4.44	0.00	7.56	239.5	3.15
ABF-Pre7-4C-W	0.00	19.14	138.4	13.83	0.00	19.10	306.0	6.24	0.00	19.02	500.6	3.80	0.00	19.15	749.7	2.55
AMG-4C-W	0.00	6.71	76.5	8.77	0.00	6.73	109.9	6.12	0.00	6.83	131.0	5.21	0.00	6.76	145.0	4.66

(a) empirical convergence results

	$L=1$						$L=2$						$L=3$						$L=4$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	1.71	121.09	0.13	2.02	43.7	4.62	1.94	35.01	0.16	1.81	35.6	5.08	2.76	11.47	0.25	1.39	34.6	4.02	4.58	4.79	0.36	1.01	34.6	2.93
ABF-Pre7	1.31	152.03	0.07	2.69	49.7	5.42	1.31	44.39	0.07	2.69	43.3	6.20	1.38	16.37	0.08	2.52	42.8	5.89	1.67	7.90	0.13	2.06	42.9	4.81
ABF-Pre7-4C	1.01	152.03	0.00	5.83	79.2	7.36	1.02	44.39	0.01	5.26	96.6	5.44	1.01	16.37	0.00	5.69	102.8	5.54	1.01	7.90	0.00	5.96	105.0	5.68
AMG-J	1.00	121.09	0.00	7.36	80.7	9.13	1.01	35.01	0.00	6.23	88.9	7.01	1.04	11.47	0.01	4.59	92.4	4.97	1.13	4.79	0.03	3.48	93.8	3.71
AMG-4C	1.03	121.09	0.01	4.88	76.5	6.37	1.04	35.01	0.01	4.72	83.6	5.65	1.05	11.47	0.01	4.39	86.9	5.05	1.08	4.79	0.02	4.01	88.2	4.55
GMG-J	1.19	121.09	0.04	3.14	83.2	3.77	1.18	35.01	0.04	3.17	92.1	3.44	1.18	11.47	0.04	3.21	95.9	3.35	1.16	4.79	0.04	3.29	97.4	3.38
V(1,1)-4C	1.03	121.09	0.01	4.88	72.2	6.76	1.04	35.01	0.01	4.72	79.3	5.96	1.05	11.47	0.01	4.39	82.6	5.31	1.08	4.79	0.02	4.01	83.8	4.79
V(0,1)-4C	1.00	121.09	-0.00	19.11	49.2	38.85	1.00	35.01	-0.00	12.55	45.9	27.31	1.00	11.47	-0.00	13.94	46.3	30.09	1.00	4.79	0.00	11.87	46.7	25.42
CMG	6.39	491.55	0.43	0.84	60.6	1.38	6.38	52.41	0.43	0.84	69.1	1.21	7.19	20.07	0.46	0.78	73.4	1.07	7.19	2.18	0.46	0.78	80.5	0.97
ABF-Pre7-W	1.00	152.03	0.00	9.62	79.4	12.12	1.00	44.39	0.00	9.58	119.4	8.02	1.00	16.37	0.00	9.58	170.2	5.63	1.00	7.90	0.00	9.58	239.5	4.00
ABF-Pre7-4C-W	1.00	152.03	0.00	14.09	138.4	10.18	1.00	44.39	0.00	14.10	306.0	4.61	1.00	16.37	0.00	14.10	500.6	2.82	1.00	7.90	0.00	14.10	749.7	1.88
AMG-4C-W	1.03	121.09	0.01	4.88	76.5	6.37	1.03	35.01	0.01	4.87	109.9	4.43	1.03	11.47	0.01	4.87	131.0	3.72	1.03	4.79	0.01	4.87	145.0	3.36

Original condition number =873.3

(b) theoretical convergence results

Table 101: HDR compression, 33×33 image

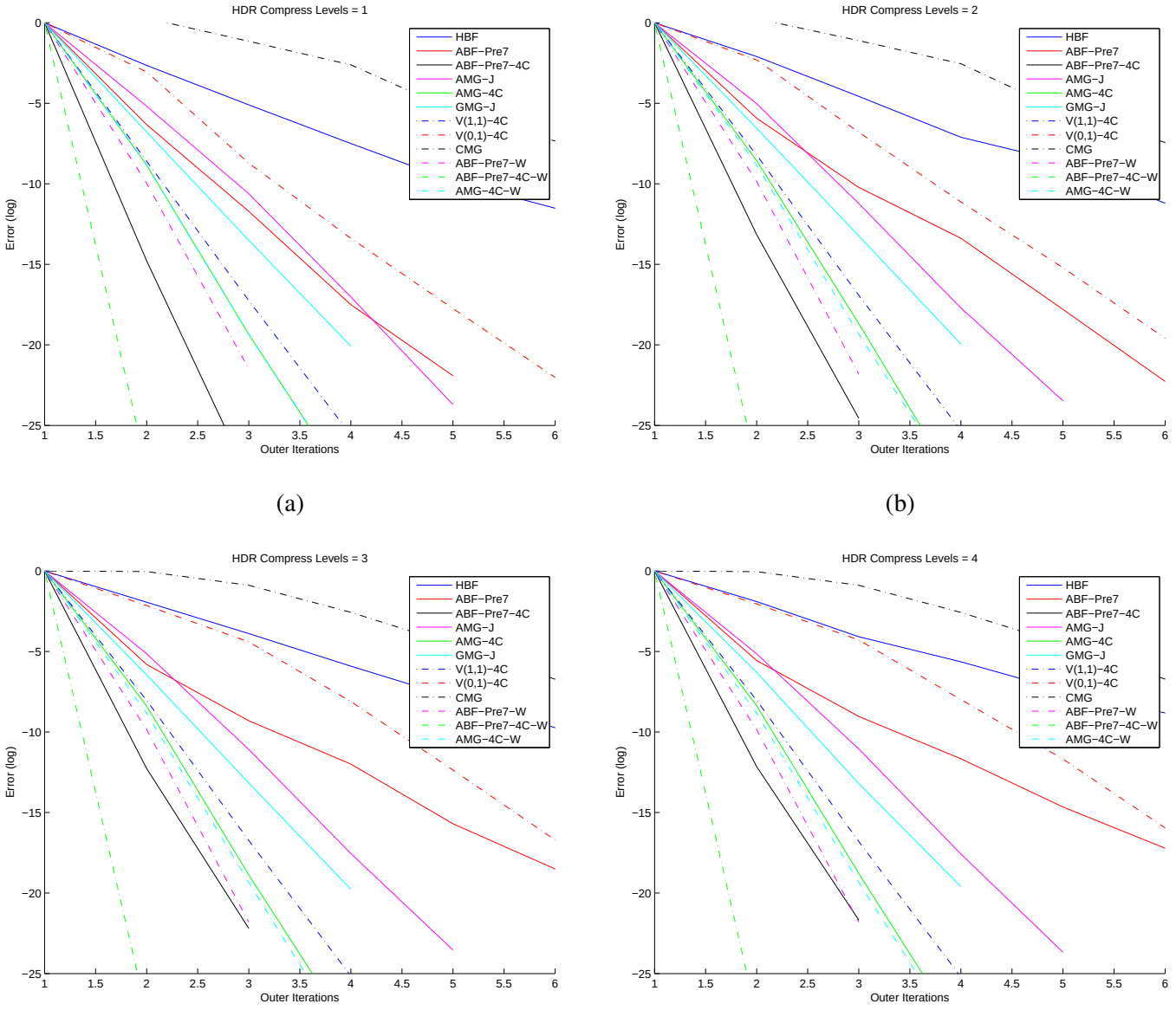


Figure 101: Log error vs. its, HDR compression, 33×33 image

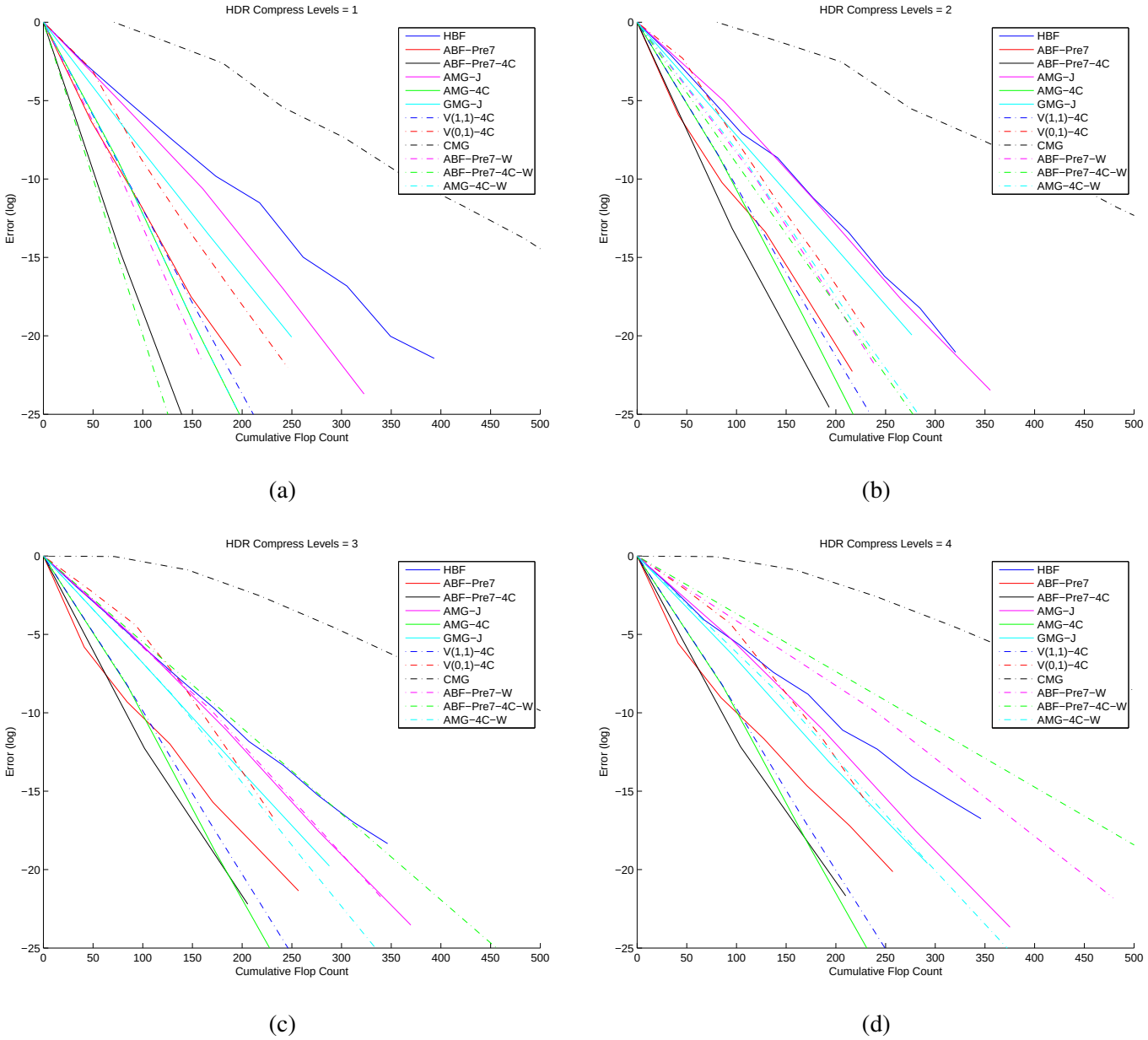
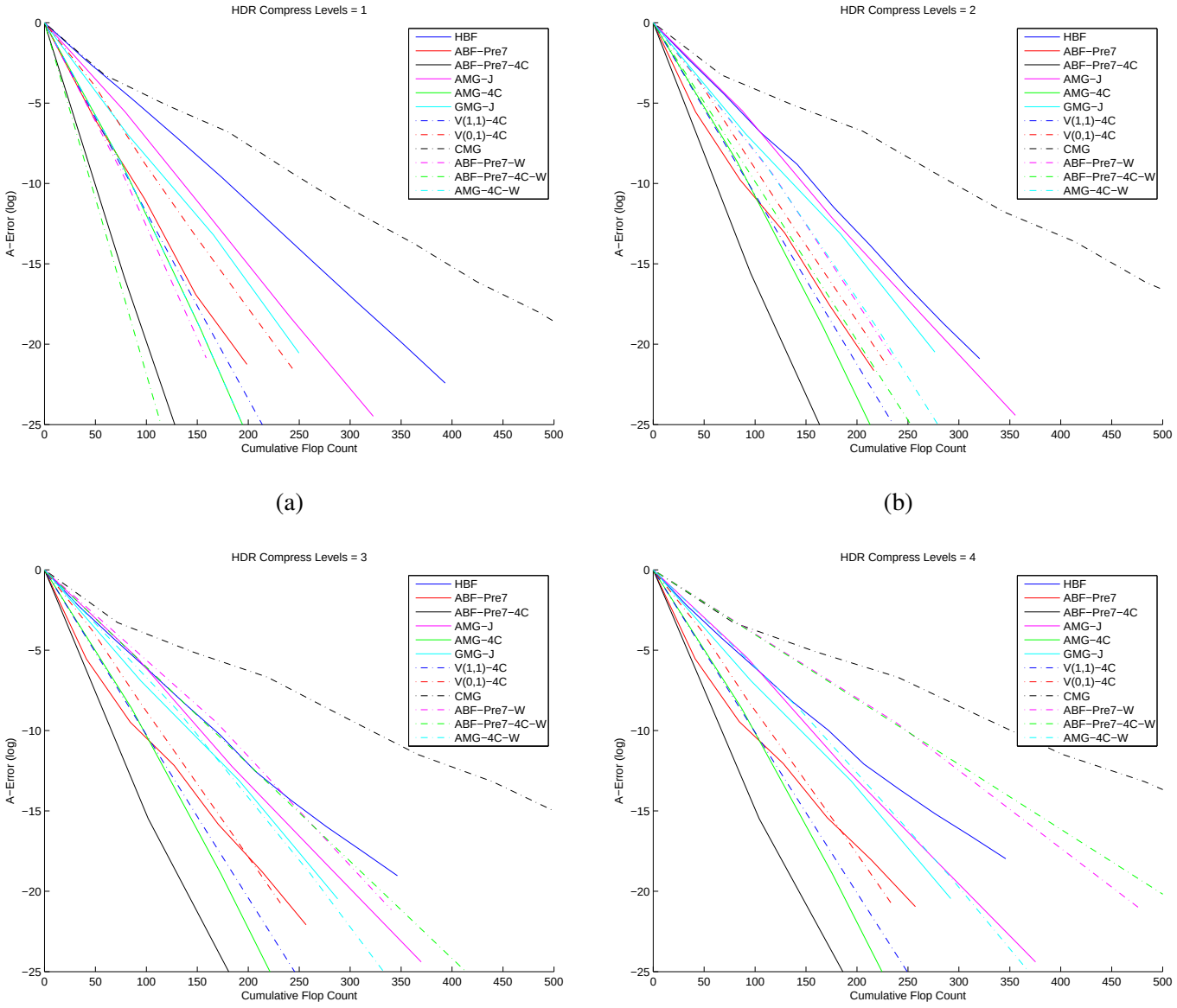


Figure 102: Log error vs. flops, HDR compression, 33×33 image



(a) (b)

(c) (d)

Figure 103: Log A-error vs. flops, HDR compression, 33×33 image

	$L=1$				$L=2$				$L=3$				$L=4$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.16	1.81	43.7	4.15	0.22	1.50	35.6	4.21	0.31	1.16	34.6	3.36	0.34	1.08	34.6	3.13
ABF-Pre7	0.02	3.71	49.7	7.46	0.06	2.84	43.3	6.56	0.10	2.29	42.8	5.36	0.11	2.24	42.9	5.22
ABF-Pre7-4C	0.00	9.29	79.2	11.73	0.00	8.44	96.6	8.73	0.00	7.46	102.8	7.26	0.00	7.23	105.0	6.89
AMG-J	0.01	4.43	80.7	5.49	0.01	4.35	88.9	4.90	0.01	4.31	92.4	4.66	0.01	4.35	93.8	4.64
AMG-4C	0.00	7.08	76.2	9.30	0.00	6.91	83.3	8.30	0.00	6.89	86.6	7.96	0.00	6.76	87.8	7.70
GMG-J	0.01	4.89	83.2	5.88	0.01	4.85	92.1	5.27	0.01	4.82	95.9	5.03	0.01	4.77	97.4	4.90
V(1,1)-4C	0.00	6.41	72.2	8.87	0.00	6.33	79.3	7.99	0.00	6.28	82.6	7.61	0.00	6.24	83.8	7.44
V(0,1)-4C	0.03	3.45	49.2	7.01	0.05	3.06	45.9	6.66	0.06	2.89	46.3	6.24	0.08	2.54	46.7	5.44
CMG	0.29	1.23	60.6	2.04	0.30	1.22	69.0	1.77	0.30	1.20	73.4	1.63	0.30	1.20	80.5	1.49
ABF-Pre7-W	0.00	7.40	79.4	9.33	0.00	7.49	119.4	6.27	0.00	7.48	170.2	4.40	0.00	7.48	239.5	3.12
ABF-Pre7-4C-W	0.00	17.78	138.4	12.84	0.00	17.78	306.0	5.81	0.00	17.78	500.6	3.55	0.00	17.78	749.7	2.37
AMG-4C-W	0.00	7.08	76.2	9.30	0.00	7.08	109.6	6.46	0.00	7.08	130.6	5.42	0.00	7.08	144.7	4.89

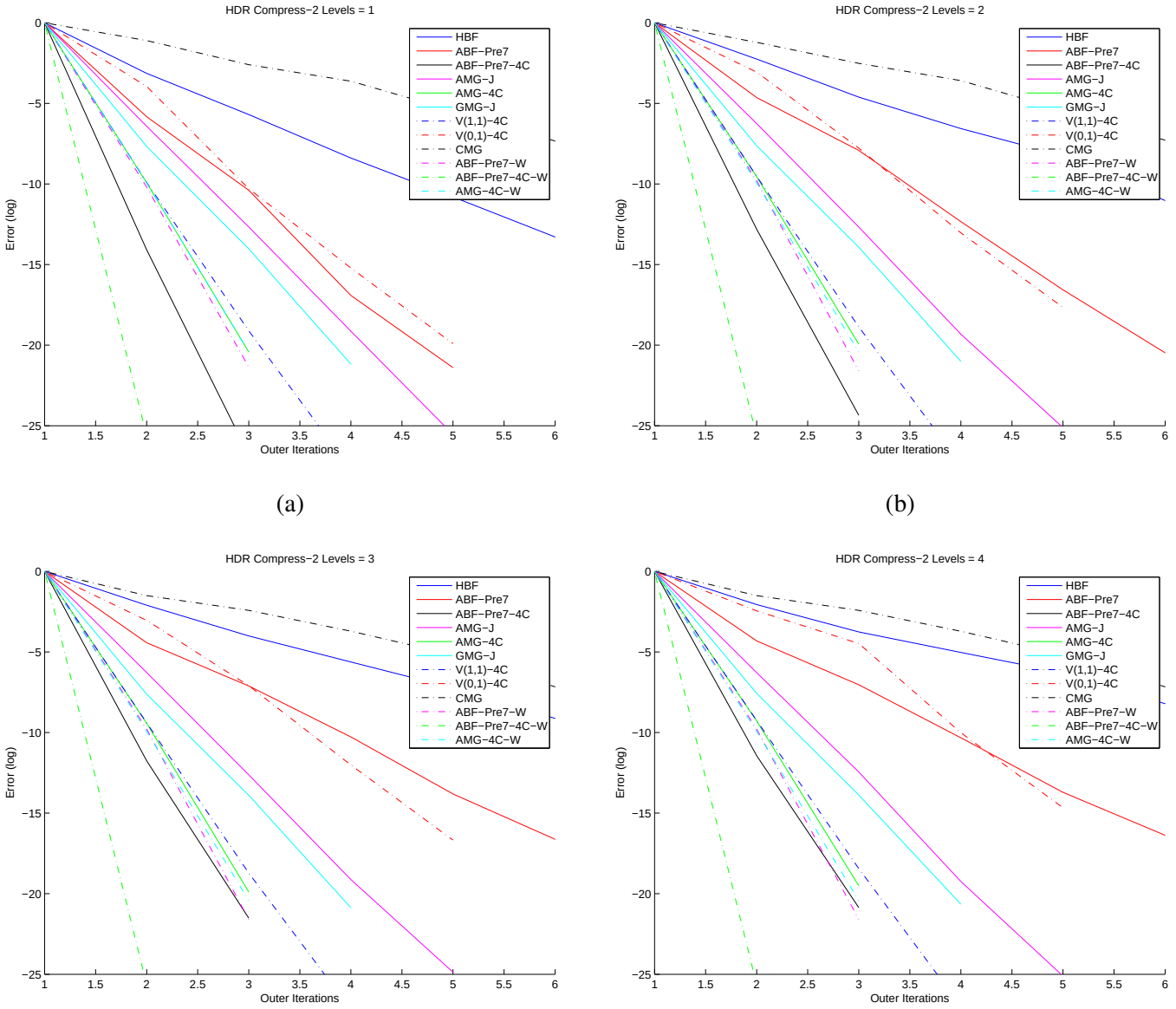
(a) empirical convergence results

	$L=1$						$L=2$						$L=3$						$L=4$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	1.71	121.09	0.13	2.02	43.7	4.62	1.94	35.01	0.16	1.81	35.6	5.08	2.76	11.47	0.25	1.39	34.6	4.02	4.58	4.79	0.36	1.01	34.6	2.93
ABF-Pre7	1.31	152.03	0.07	2.69	49.7	5.42	1.31	44.39	0.07	2.69	43.3	6.20	1.38	16.37	0.08	2.52	42.8	5.89	1.67	7.90	0.13	2.06	42.9	4.81
ABF-Pre7-4C	1.01	152.03	0.00	5.83	79.2	7.36	1.02	44.39	0.01	5.26	96.6	5.44	1.01	16.37	0.00	5.69	102.8	5.54	1.01	7.90	0.00	5.96	105.0	5.68
AMG-J	1.00	121.09	0.00	7.36	80.7	9.13	1.01	35.01	0.00	6.23	88.9	7.01	1.04	11.47	0.01	4.59	92.4	4.97	1.13	4.79	0.03	3.48	93.8	3.71
AMG-4C	1.03	121.09	0.01	4.88	76.2	6.40	1.04	35.01	0.01	4.72	83.3	5.67	1.05	11.47	0.01	4.39	86.6	5.07	1.08	4.79	0.02	4.01	87.8	4.57
GMG-J	1.19	121.09	0.04	3.14	83.2	3.77	1.18	35.01	0.04	3.17	92.1	3.44	1.18	11.47	0.04	3.21	95.9	3.35	1.16	4.79	0.04	3.29	97.4	3.38
V(1,1)-4C	1.03	121.09	0.01	4.88	72.2	6.76	1.04	35.01	0.01	4.72	79.3	5.96	1.05	11.47	0.01	4.39	82.6	5.31	1.08	4.79	0.02	4.01	83.8	4.79
V(0,1)-4C	1.00	121.09	-0.00	19.11	49.2	38.85	1.00	35.01	-0.00	12.55	45.9	27.31	1.00	11.47	-0.00	13.94	46.3	30.09	1.00	4.79	0.00	11.87	46.7	25.42
CMG	6.39	491.55	0.43	0.84	60.6	1.38	6.38	52.41	0.43	0.84	69.0	1.21	7.19	20.07	0.46	0.78	73.4	1.07	7.19	2.18	0.46	0.78	80.5	0.97
ABF-Pre7-W	1.00	152.03	0.00	9.62	79.4	12.12	1.00	44.39	0.00	9.58	119.4	8.02	1.00	16.37	0.00	9.58	170.2	5.63	1.00	7.90	0.00	9.58	239.5	4.00
ABF-Pre7-4C-W	1.00	152.03	0.00	14.09	138.4	10.18	1.00	44.39	0.00	14.10	306.0	4.61	1.00	16.37	0.00	14.10	500.6	2.82	1.00	7.90	0.00	14.10	749.7	1.88
AMG-4C-W	1.03	121.09	0.01	4.88	76.2	6.40	1.03	35.01	0.01	4.87	109.6	4.44	1.03	11.47	0.01	4.87	130.6	3.73	1.03	4.79	0.01	4.87	144.7	3.37

Original condition number =873.3

(b) theoretical convergence results

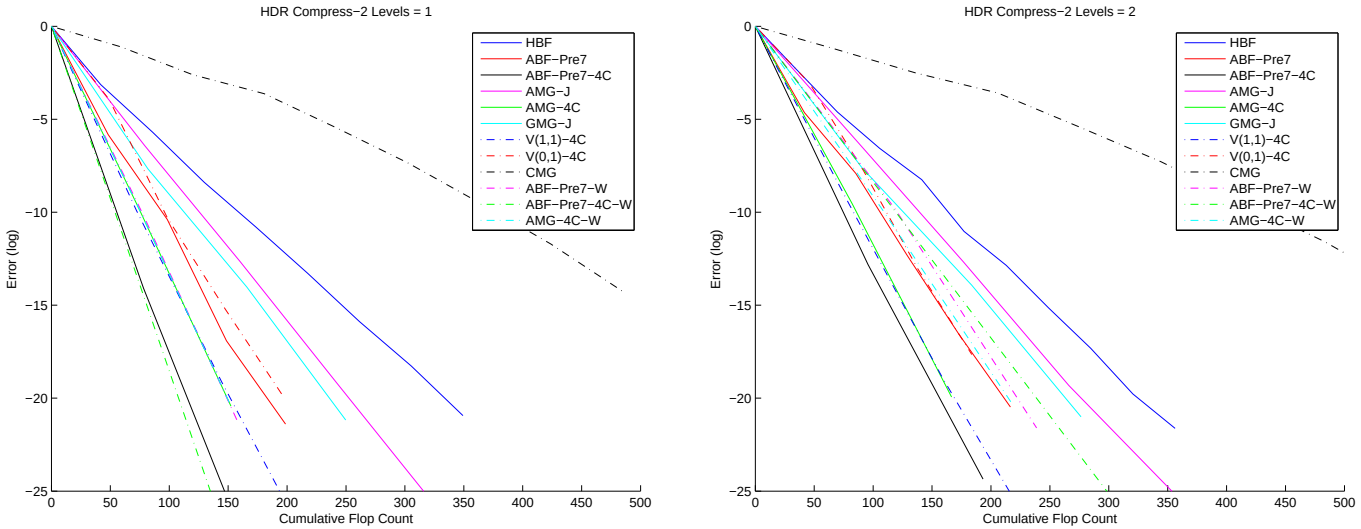
Table 102: HDR compression, 33×33 image



(a) (b)

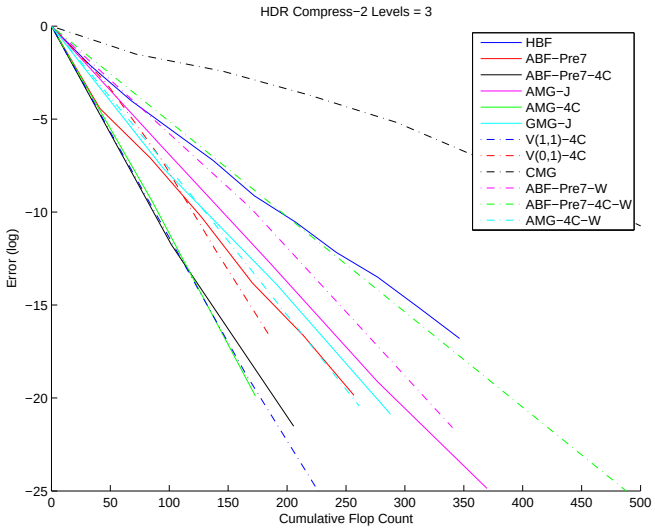
(c) (d)

Figure 104: Log error vs. its, HDR compression, 33×33 image

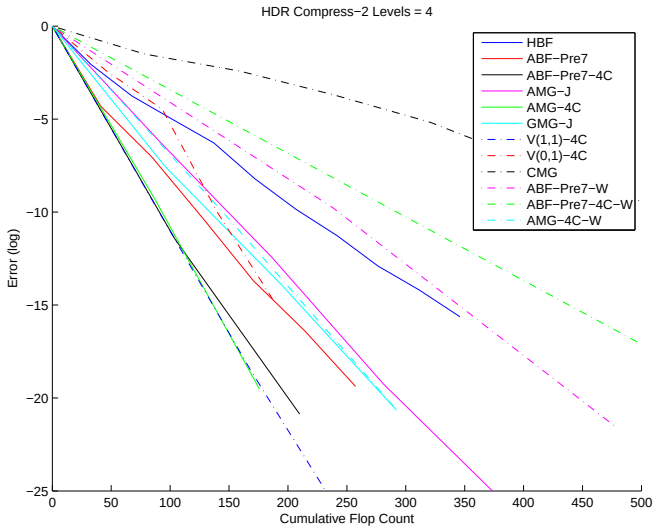


(a)

(b)

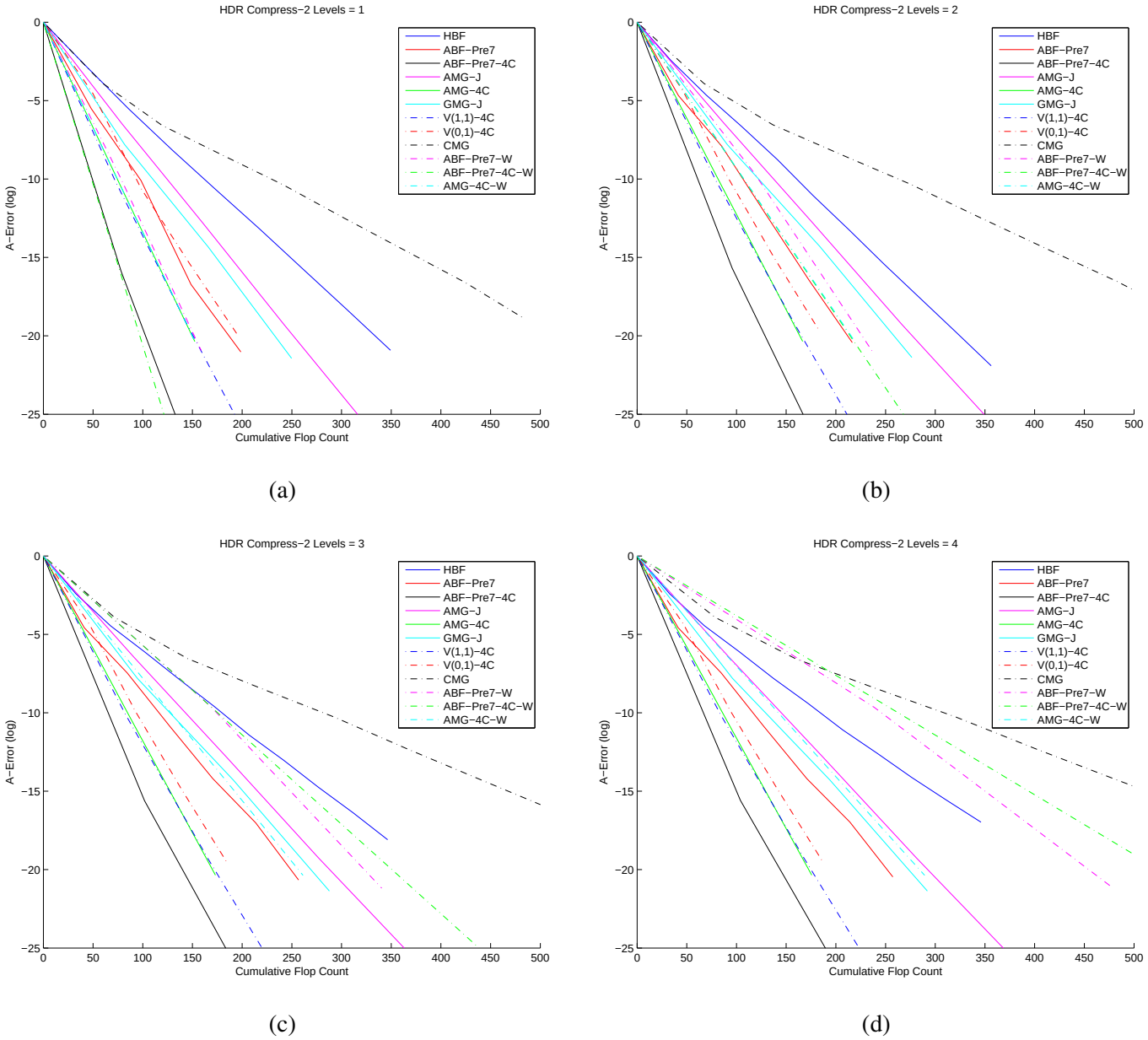


(c)



(d)

Figure 105: Log error vs. flops, HDR compression, 33×33 image



(a) (b)

(c) (d)

Figure 106: Log A-error vs. flops, HDR compression, 33×33 image

	$L=5$				$L=6$				$L=7$				$L=8$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.38	0.98	33.2	2.95	0.42	0.88	33.1	2.64	0.46	0.78	33.1	2.35	0.50	0.69	33.1	2.09
ABF-Pre7	0.19	1.68	41.8	4.01	0.21	1.56	41.8	3.74	0.29	1.25	41.8	2.98	0.31	1.18	41.8	2.81
ABF-Pre7-4C	0.00	6.73	100.9	6.67	0.00	6.01	101.0	5.95	0.00	5.64	101.0	5.59	0.00	5.60	101.0	5.55
AMG-J	0.03	3.67	90.5	4.06	0.04	3.19	90.5	3.52	0.05	2.96	90.5	3.27	0.06	2.89	90.5	3.20
AMG-4C	0.01	4.74	85.0	5.58	0.01	4.20	85.0	4.95	0.02	3.92	85.0	4.61	0.02	3.88	85.0	4.56
GMG-J	0.02	4.01	94.1	4.26	0.03	3.57	94.1	3.79	0.04	3.33	94.2	3.53	0.04	3.27	94.2	3.48
V(1,1)-4C	0.01	4.37	80.6	5.41	0.02	3.81	80.7	4.73	0.02	3.69	80.7	4.58	0.02	3.71	80.7	4.59
V(0,1)-4C	0.11	2.25	44.7	5.04	0.16	1.82	44.7	4.07	0.24	1.44	44.7	3.21	0.29	1.23	44.7	2.74
CMG	0.42	0.87	72.6	1.20	0.42	0.87	73.1	1.19	0.42	0.86	73.4	1.17	0.42	0.86	74.0	1.16
ABF-Pre7-W	0.00	8.80	252.2	3.49	0.00	8.80	285.8	3.08	0.00	8.80	323.1	2.72	0.00	8.80	363.5	2.42
ABF-Pre7-4C-W	0.00	21.38	791.2	2.70	0.00	21.38	944.8	2.26	0.00	21.38	1102.1	1.94	0.00	21.38	1262.5	1.69
AMG-4C-W	0.00	7.46	136.7	5.46	0.00	7.46	138.6	5.38	0.00	7.46	139.7	5.34	0.00	7.46	140.2	5.32

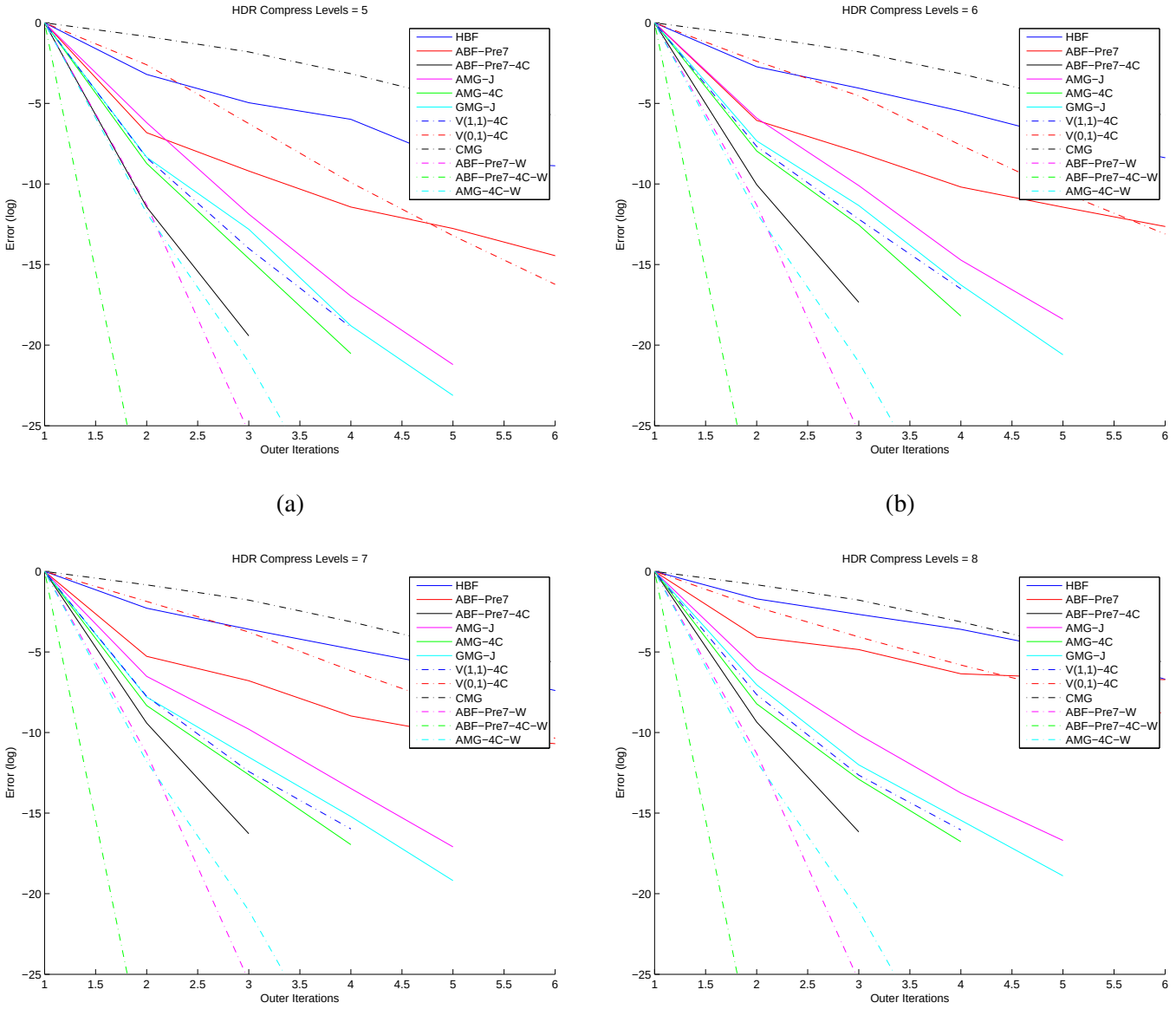
(a) empirical convergence results

	$L=5$						$L=6$						$L=7$						$L=8$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	4.23	148.32	0.35	1.06	33.2	3.20	6.49	38.70	0.44	0.83	33.1	2.50	8.49	11.43	0.49	0.72	33.1	2.16	9.71	4.27	0.51	0.67	33.1	2.01
ABF-Pre7	1.45	136.38	0.09	2.38	41.8	5.70	1.53	37.42	0.11	2.25	41.8	5.37	1.78	15.22	0.14	1.94	41.8	4.64	2.27	5.61	0.20	1.60	41.8	3.82
ABF-Pre7-4C	1.11	136.38	0.03	3.65	100.9	3.62	1.10	37.42	0.02	3.77	101.0	3.74	1.08	15.22	0.02	3.98	101.0	3.94	1.07	5.61	0.02	4.05	101.0	4.01
AMG-J	1.06	148.32	0.02	4.18	90.5	4.62	1.22	38.70	0.05	3.02	90.5	3.33	1.46	11.43	0.09	2.36	90.5	2.61	1.67	4.27	0.13	2.06	90.5	2.27
AMG-4C	1.02	148.32	0.00	5.59	85.0	6.57	1.05	38.70	0.01	4.34	85.0	5.10	1.18	11.43	0.04	3.16	85.0	3.72	1.27	4.27	0.06	2.81	85.0	3.31
GMG-J	1.23	148.32	0.05	2.96	94.1	3.15	1.30	38.70	0.07	2.73	94.1	2.90	1.44	11.43	0.09	2.40	94.2	2.54	1.56	4.27	0.11	2.19	94.2	2.33
V(1,1)-4C	1.02	148.32	0.00	5.59	80.6	6.93	1.05	38.70	0.01	4.34	80.7	5.38	1.18	11.43	0.04	3.16	80.7	3.92	1.27	4.27	0.06	2.81	80.7	3.49
V(0,1)-4C	1.36	148.32	0.08	2.56	44.7	5.72	1.70	38.70	0.13	2.03	44.7	4.54	2.05	11.43	0.18	1.73	44.7	3.87	2.26	4.27	0.20	1.60	44.7	3.59
CMG	9.82	480.79	0.52	0.66	72.6	0.91	9.82	55.18	0.52	0.66	73.1	0.90	10.20	22.63	0.52	0.65	73.4	0.88	10.21	2.50	0.52	0.65	74.0	0.88
ABF-Pre7-W	-	-	-	-	252.2	-	-	-	-	-	285.8	-	-	-	-	-	323.1	-	-	-	-	-	363.5	-
ABF-Pre7-4C-W	-	-	-	-	791.2	-	-	-	-	-	944.8	-	-	-	-	-	1102.1	-	-	-	-	-	1262.5	-
AMG-4C-W	-	-	-	-	136.7	-	-	-	-	-	138.6	-	-	-	-	-	139.7	-	-	-	-	-	140.2	-

Original condition number =179246.4

(b) theoretical convergence results

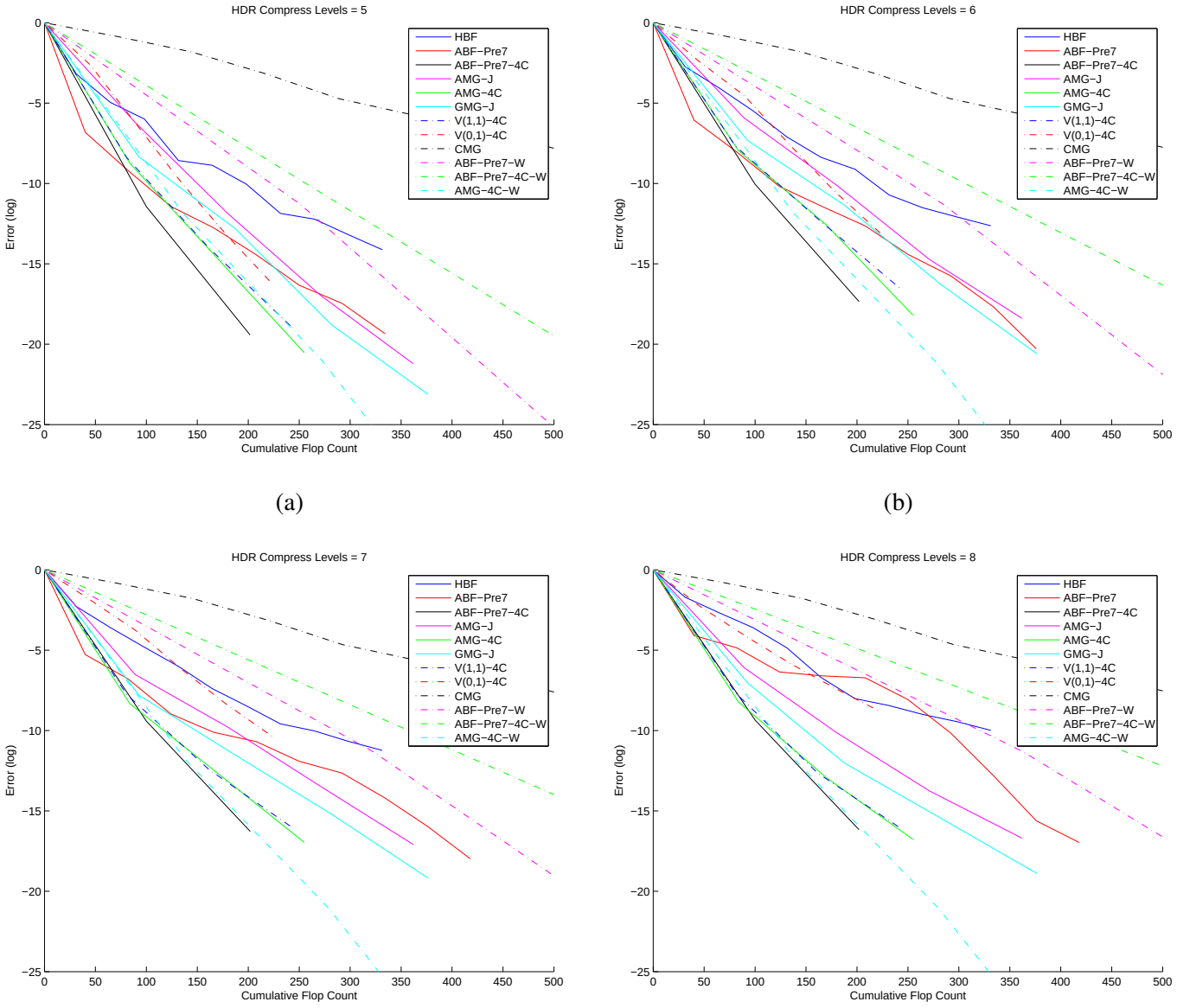
Table 103: HDR compression, 512×512 image



(a) (b)

(c) (d)

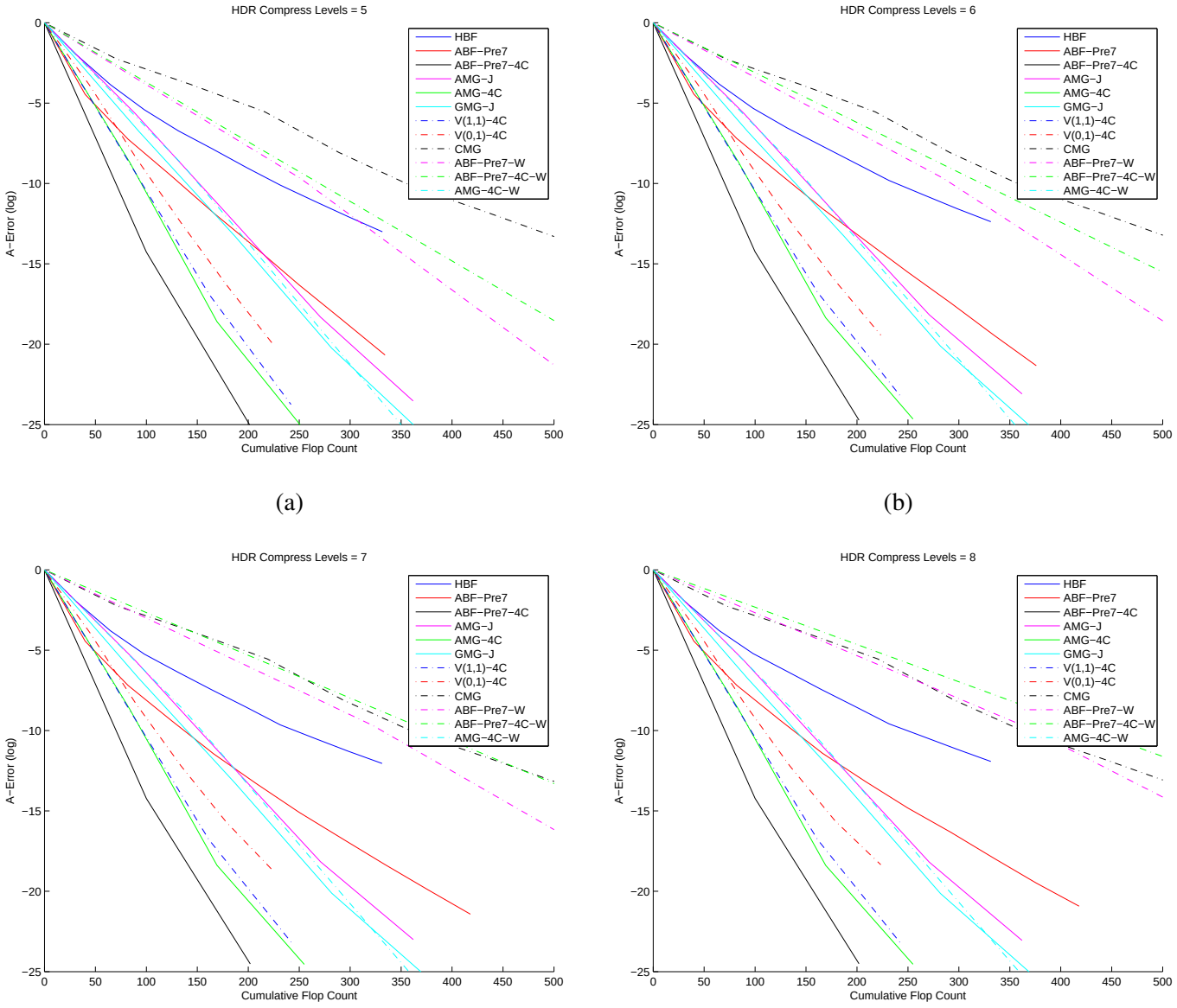
Figure 107: Log error vs. its, HDR compression, 512×512 image



(a) (b)

(c) (d)

Figure 108: Log error vs. flops, HDR compression, 512×512 image



(a) (b)

(c) (d)

Figure 109: Log A-error vs. flops, HDR compression, 512×512 image

Algorithm	$L=4$				$L=5$				$L=6$				$L=7$			
	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.32	1.13	33.4	3.38	0.36	1.01	33.2	3.06	0.42	0.87	33.1	2.62	0.47	0.77	33.1	2.31
ABF-Pre7	0.10	2.28	41.9	5.43	0.18	1.71	41.8	4.09	0.25	1.37	41.8	3.29	0.30	1.20	41.8	2.88
ABF-Pre7-4C	0.00	7.44	100.8	7.38	0.00	6.59	100.9	6.53	0.00	5.57	101.0	5.51	0.01	4.79	101.0	4.75
AMG-J	0.02	4.06	90.4	4.49	0.02	3.80	90.5	4.20	0.03	3.67	90.5	4.06	0.03	3.38	90.5	3.73
AMG-4C	0.00	5.58	85.0	6.57	0.01	5.23	85.0	6.16	0.01	4.88	85.0	5.75	0.01	4.21	85.0	4.95
GMG-J	0.01	4.94	93.9	5.26	0.01	4.65	94.0	4.95	0.01	4.50	94.0	4.79	0.02	3.92	94.0	4.17
V(1,1)-4C	0.00	5.37	80.6	6.66	0.01	5.04	80.6	6.25	0.01	4.61	80.7	5.72	0.02	3.97	80.7	4.93
V(0,1)-4C	0.07	2.60	44.8	5.80	0.10	2.30	44.7	5.16	0.13	2.03	44.7	4.55	0.15	1.91	44.7	4.28
CMG	0.36	1.03	74.1	1.39	0.39	0.93	72.7	1.29	0.39	0.93	73.3	1.27	0.40	0.92	73.5	1.25
ABF-Pre7-W	0.00	8.31	223.9	3.71	0.00	8.31	254.5	3.26	0.00	8.31	287.9	2.89	0.00	8.31	324.1	2.56
ABF-Pre7-4C-W	0.00	21.02	642.9	3.27	0.00	21.02	793.5	2.65	0.00	21.02	946.9	2.22	0.00	21.02	1103.1	1.91
AMG-4C-W	0.00	7.21	133.5	5.40	0.00	7.20	136.8	5.27	0.00	7.21	138.6	5.20	0.00	7.21	139.7	5.16

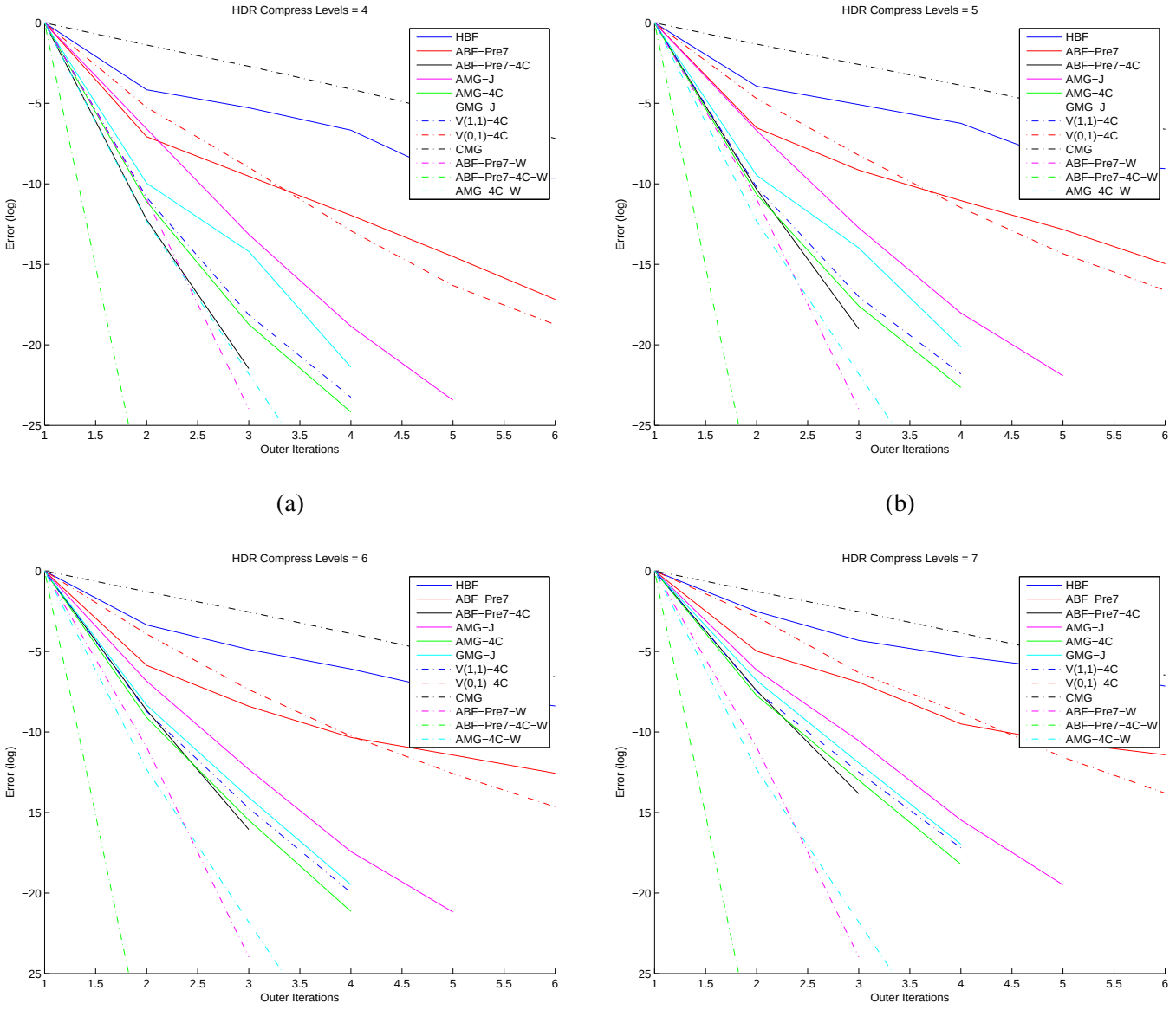
(a) empirical convergence results

Algorithm	$L=4$						$L=5$						$L=6$						$L=7$					
	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	3.08	1306.74	0.27	1.30	33.4	3.88	4.24	330.49	0.35	1.06	33.2	3.20	6.54	83.49	0.44	0.83	33.1	2.49	8.62	21.47	0.49	0.71	33.1	2.14
ABF-Pre7	1.42	1195.26	0.09	2.45	41.9	5.83	1.46	303.74	0.09	2.36	41.8	5.65	1.55	78.98	0.11	2.21	41.8	5.30	1.80	23.10	0.15	1.92	41.8	4.61
ABF-Pre7-4C	1.11	1195.26	0.03	3.61	100.8	3.59	1.12	303.74	0.03	3.57	100.9	3.54	1.11	78.98	0.03	3.61	101.0	3.57	1.11	23.10	0.03	3.69	101.0	3.65
AMG-J	1.01	1306.74	0.00	5.70	90.4	6.31	1.06	330.49	0.02	4.16	90.5	4.60	1.22	83.49	0.05	3.00	90.5	3.32	1.46	21.47	0.09	2.36	90.5	2.61
AMG-4C	1.00	1306.74	0.00	6.96	85.0	8.20	1.02	330.49	0.00	5.57	85.0	6.56	1.06	83.49	0.01	4.31	85.0	5.07	1.19	21.47	0.04	3.14	85.0	3.69
GMG-J	1.21	1306.74	0.05	3.06	93.9	3.25	1.23	330.49	0.05	2.97	94.0	3.16	1.30	83.49	0.07	2.73	94.0	2.90	1.45	21.47	0.09	2.38	94.0	2.54
V(1,1)-4C	1.00	1306.74	0.00	6.96	80.6	8.64	1.02	330.49	0.00	5.57	80.6	6.91	1.06	83.49	0.01	4.31	80.7	5.35	1.19	21.47	0.04	3.14	80.7	3.89
V(0,1)-4C	1.21	1306.74	0.05	3.03	44.8	6.77	1.37	330.49	0.08	2.55	44.7	5.70	1.71	83.49	0.13	2.02	44.7	4.52	2.07	21.47	0.18	1.72	44.7	3.85
CMG	8.52	1814.41	0.49	0.71	74.1	0.96	9.83	1079.78	0.52	0.66	72.7	0.91	9.83	118.11	0.52	0.66	73.3	0.90	10.22	52.92	0.52	0.65	73.5	0.88
ABF-Pre7-W	-	-	-	-	223.9	-	-	-	-	-	254.5	-	-	-	-	-	287.9	-	-	-	-	-	324.1	-
ABF-Pre7-4C-W	-	-	-	-	642.9	-	-	-	-	-	793.5	-	-	-	-	-	946.9	-	-	-	-	-	1103.1	-
AMG-4C-W	-	-	-	-	133.5	-	-	-	-	-	136.8	-	-	-	-	-	138.6	-	-	-	-	-	139.7	-

Original condition number =178771.4

(b) theoretical convergence results

Table 104: HDR compression, 768×512 image



(a) (b)

(c) (d)

Figure 110: Log error vs. its, HDR compression, 768×512 image

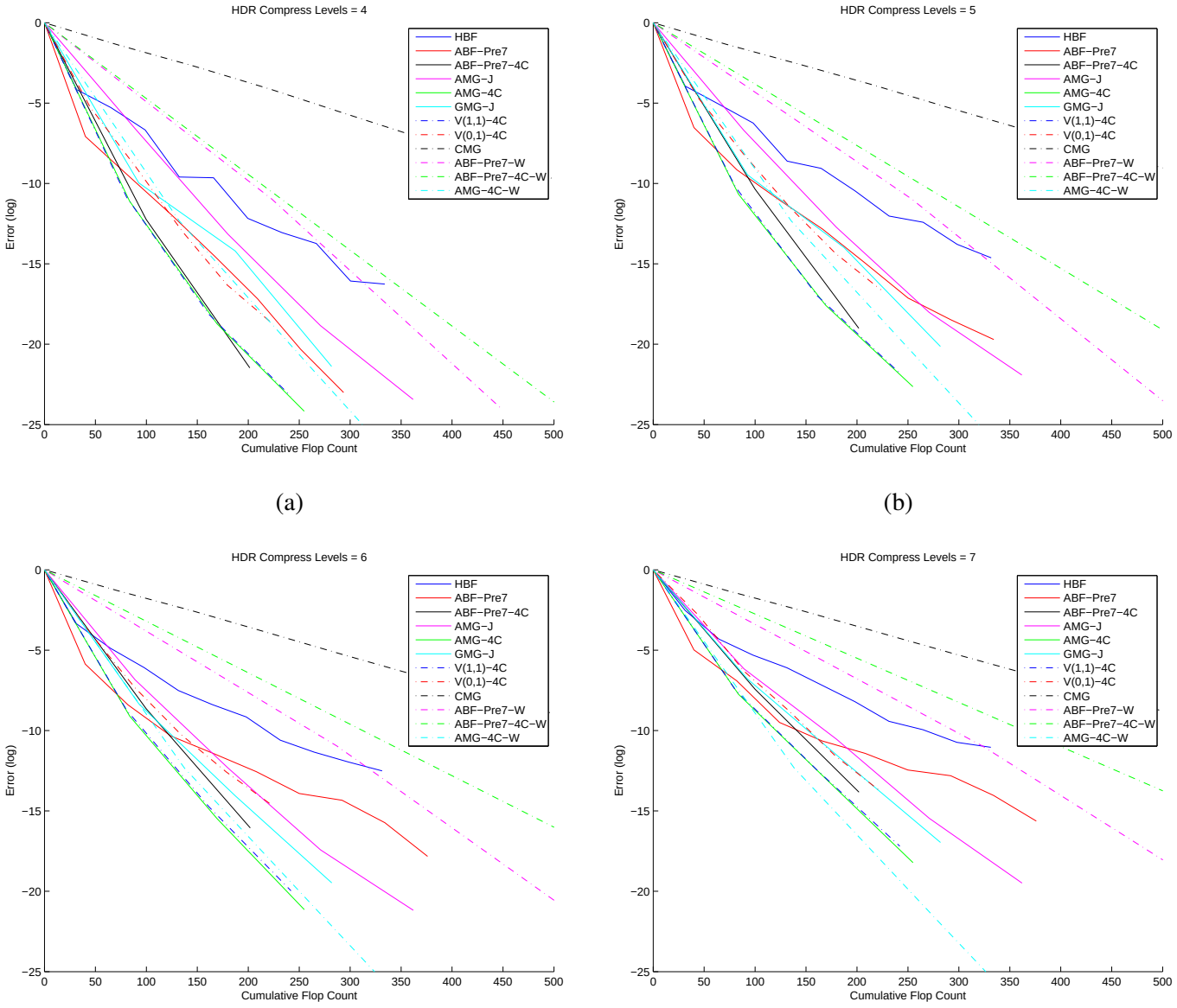
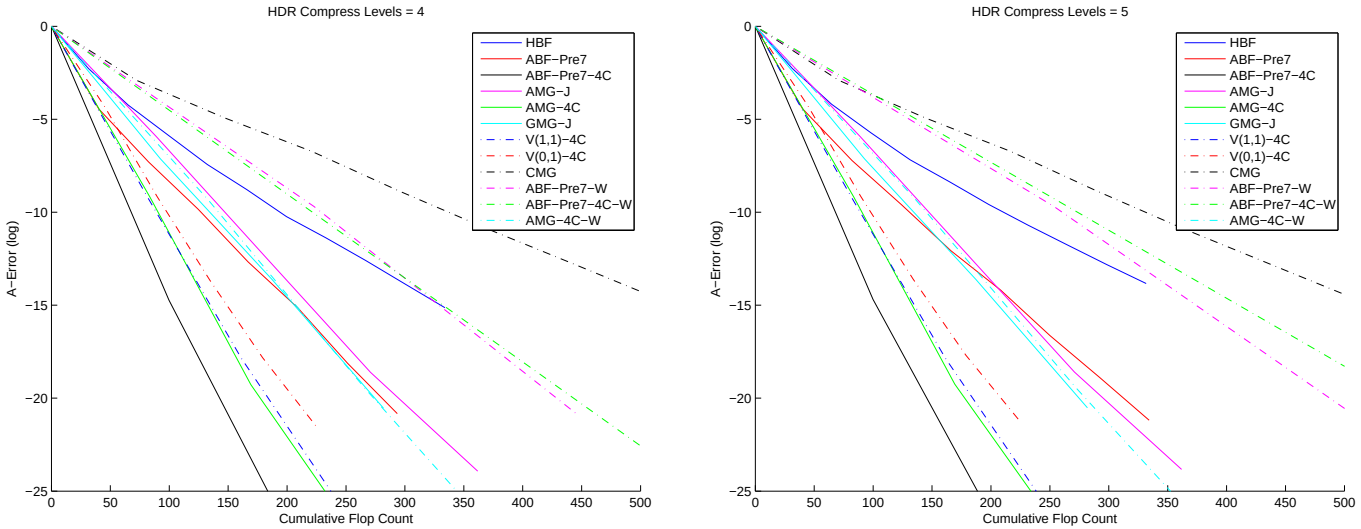
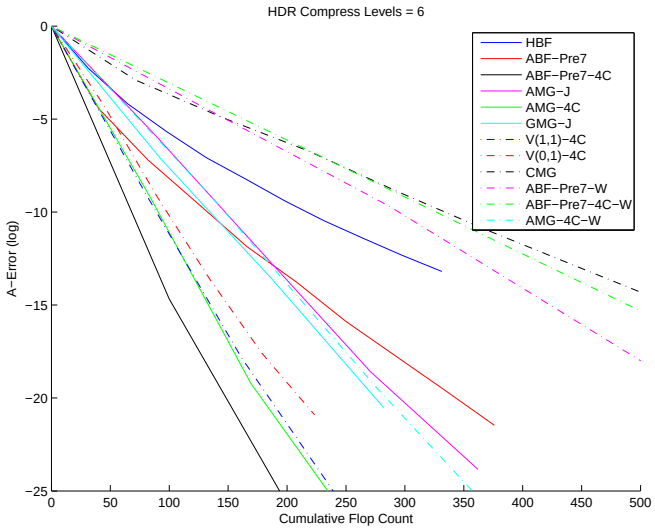


Figure 111: Log error vs. flops, HDR compression, 768×512 image

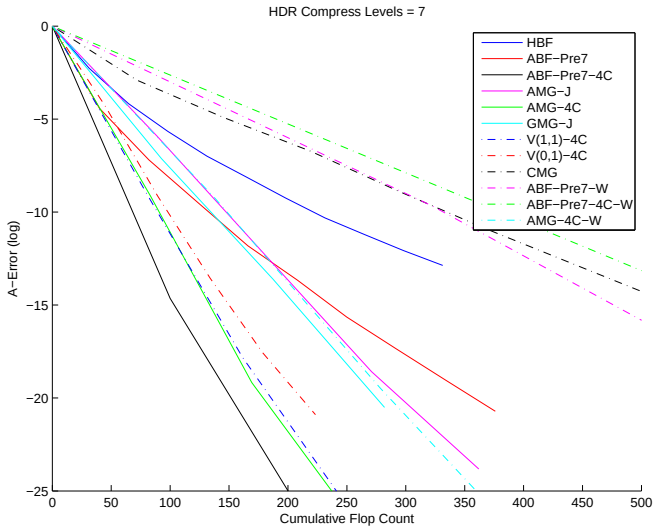


(a)

(b)



(c)



(d)

Figure 112: Log A-error vs. flops, HDR compression, 768×512 image

	$L=7$				$L=8$				$L=9$				$L=10$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.38	0.98	33.1	2.95	0.41	0.89	33.1	2.68	0.44	0.82	33.1	2.46	0.48	0.74	33.1	2.23
ABF-Pre7	0.25	1.40	41.8	3.35	0.30	1.21	41.8	2.88	0.33	1.12	41.8	2.67	0.34	1.09	41.8	2.61
ABF-Pre7-4C	0.00	6.46	101.0	6.39	0.00	6.04	101.0	5.97	0.00	5.87	101.0	5.81	0.00	5.81	101.0	5.75
AMG-J	0.02	3.72	90.5	4.11	0.02	3.89	90.5	4.29	0.05	3.05	90.5	3.37	0.07	2.65	90.5	2.93
AMG-4C	0.00	5.35	85.0	6.29	0.01	4.47	85.0	5.26	0.02	3.83	85.0	4.50	0.03	3.54	85.0	4.16
GMG-J	0.02	4.19	94.2	4.44	0.02	4.16	94.2	4.41	0.04	3.34	94.2	3.55	0.05	3.03	94.2	3.21
V(1,1)-4C	0.01	5.14	80.7	6.37	0.02	4.15	80.7	5.14	0.03	3.55	80.7	4.40	0.04	3.27	80.7	4.06
V(0,1)-4C	0.09	2.38	44.7	5.33	0.09	2.42	44.7	5.41	0.14	1.96	44.7	4.38	0.21	1.58	44.7	3.55

(a) empirical convergence results

Table 105: HDR compression, 2048×1365 image

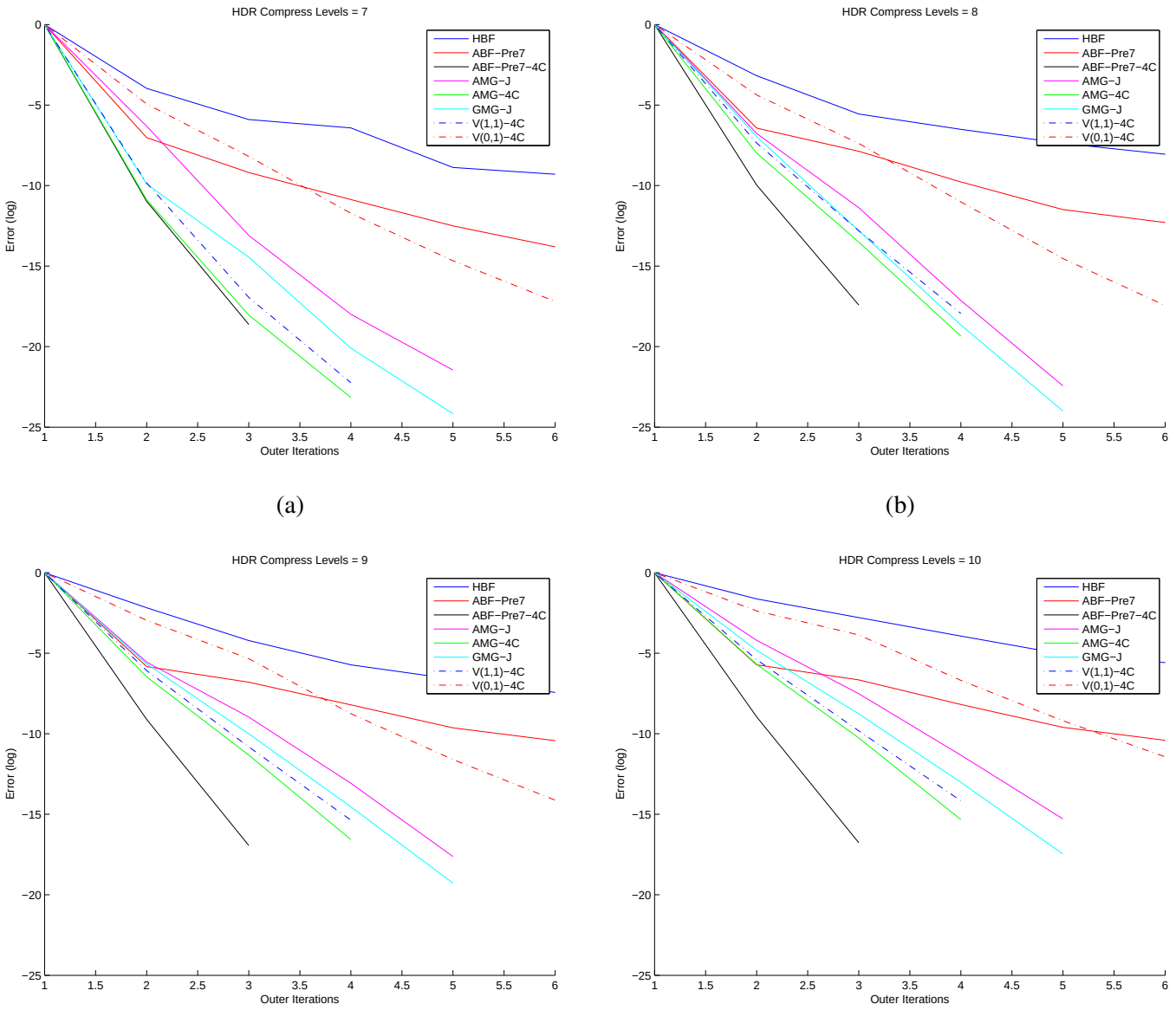
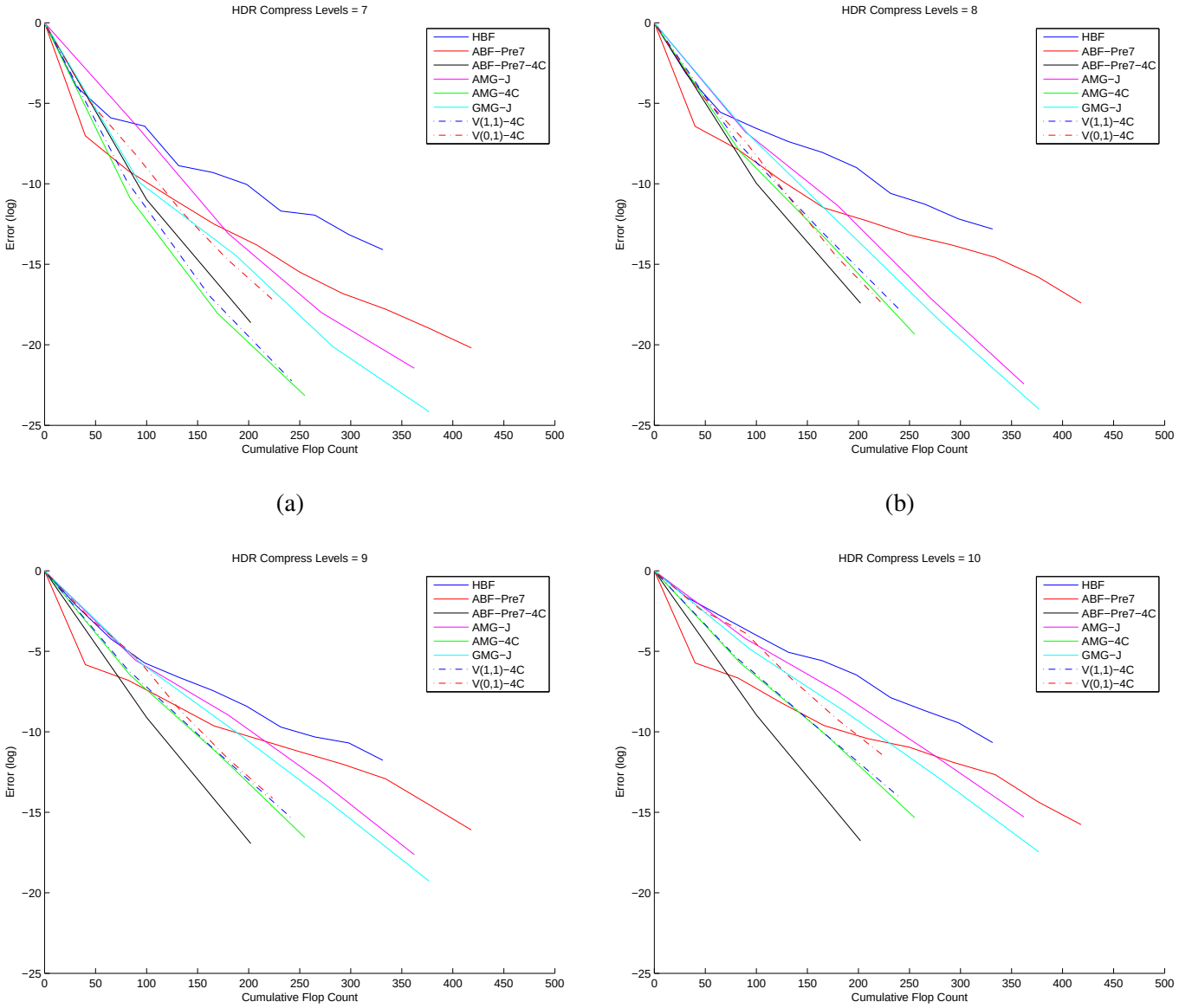


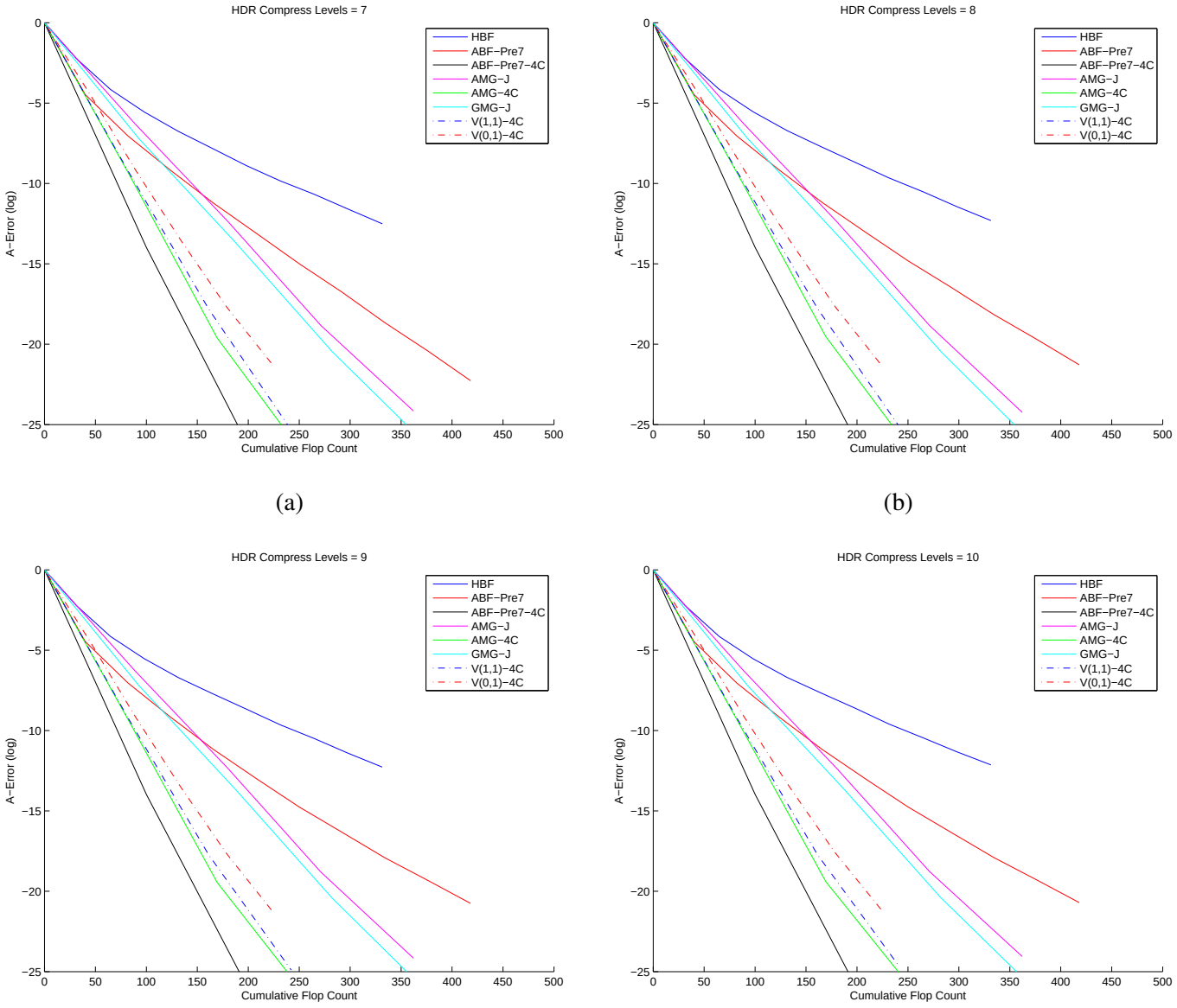
Figure 113: Log error vs. its, HDR compression, 2048×1365 image



(a) (b)

(c) (d)

Figure 114: Log error vs. flops, HDR compression, 2048×1365 image



(a) (b)

(c) (d)

Figure 115: Log A-error vs. flops, HDR compression, 2048×1365 image

	$L=1$				$L=2$				$L=3$				$L=4$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.18	1.74	33.8	5.16	0.18	1.70	33.3	5.11	0.24	1.45	33.6	4.30	0.30	1.20	33.9	3.54
ABF-Pre7	0.02	4.02	40.0	10.07	0.05	3.08	41.2	7.49	0.08	2.53	41.9	6.04	0.10	2.28	42.3	5.39
ABF-Pre7-4C	0.00	9.91	69.6	14.24	0.00	9.32	94.7	9.85	0.00	8.59	102.1	8.41	0.00	8.38	104.6	8.01
AMG-J	0.01	4.80	70.1	6.85	0.01	4.70	85.6	5.48	0.01	4.70	90.5	5.19	0.01	4.43	92.1	4.82
AMG-4C	0.00	7.54	67.2	11.21	0.00	7.32	82.1	8.92	0.00	7.21	86.7	8.31	0.00	7.28	88.3	8.25
GMG-J	0.01	5.17	72.3	7.16	0.01	5.27	88.4	5.97	0.01	5.16	93.4	5.53	0.01	4.84	95.1	5.09
V(1,1)-4C	0.00	6.80	62.7	10.84	0.00	6.60	77.6	8.50	0.00	6.40	82.2	7.78	0.00	6.42	83.8	7.67
V(0,1)-4C	0.03	3.46	39.7	8.71	0.04	3.32	44.2	7.50	0.04	3.19	46.0	6.93	0.05	2.94	46.6	6.30
CMG	0.20	1.60	60.6	2.63	0.21	1.57	69.1	2.28	0.21	1.54	73.5	2.10	0.21	1.54	80.5	1.92
ABF-Pre7-W	0.00	7.69	60.3	12.75	0.00	7.78	104.9	7.41	0.00	7.78	157.9	4.93	0.00	7.78	227.8	3.41
ABF-Pre7-4C-W	0.00	17.82	119.9	14.87	0.00	17.83	292.1	6.10	0.00	17.83	488.7	3.65	0.00	17.83	738.5	2.41
AMG-4C-W	0.00	7.54	67.2	11.21	0.00	7.53	106.7	7.06	0.00	7.53	129.8	5.81	0.00	7.53	144.6	5.21

(a) empirical convergence results

	$L=1$						$L=2$						$L=3$						$L=4$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	1.71	1061.85	0.13	2.02	33.8	5.97	1.95	287.05	0.17	1.80	33.3	5.40	2.87	81.54	0.26	1.36	33.6	4.03	5.06	25.85	0.38	0.96	33.9	2.82
ABF-Pre7	1.31	1322.81	0.07	2.69	40.0	6.73	1.31	354.47	0.07	2.68	41.2	6.52	1.38	105.86	0.08	2.53	41.9	6.04	1.63	32.54	0.12	2.11	42.3	4.98
ABF-Pre7-4C	1.01	1322.81	0.00	5.82	69.6	8.36	1.02	354.47	0.01	5.25	94.7	5.54	1.01	105.86	0.00	5.69	102.1	5.57	1.01	32.54	0.00	5.95	104.6	5.69
AMG-J	1.00	1062.24	0.00	7.33	70.1	10.46	1.01	287.19	0.00	6.23	85.6	7.28	1.04	81.74	0.01	4.62	90.5	5.11	1.12	26.21	0.03	3.53	92.1	3.83
AMG-4C	1.03	1062.24	0.01	4.88	67.2	7.26	1.04	287.19	0.01	4.75	82.1	5.79	1.05	81.74	0.01	4.45	86.7	5.13	1.07	26.21	0.02	4.12	88.3	4.67
GMG-J	1.19	1061.85	0.04	3.14	72.3	4.35	1.18	287.05	0.04	3.18	88.4	3.60	1.17	81.54	0.04	3.26	93.4	3.49	1.14	25.85	0.03	3.43	95.1	3.61
V(1,1)-4C	1.03	1062.24	0.01	4.88	62.7	7.78	1.04	287.19	0.01	4.75	77.6	6.13	1.05	81.74	0.01	4.45	82.2	5.41	1.07	26.21	0.02	4.12	83.8	4.92
V(0,1)-4C	1.12	1062.24	0.03	3.57	39.7	8.99	1.15	287.19	0.04	3.35	44.2	7.56	1.17	81.74	0.04	3.26	46.0	7.09	1.18	26.21	0.04	3.16	46.6	6.78
CMG	5.89	4953.66	0.42	0.88	60.6	1.45	5.88	1875.04	0.42	0.88	69.1	1.27	6.59	1032.83	0.44	0.82	73.5	1.12	6.59	250.46	0.44	0.82	80.5	1.02
ABF-Pre7-W	1.00	1322.81	0.00	9.70	60.3	16.08	1.00	354.47	0.00	9.67	104.9	9.21	1.00	105.86	0.00	9.67	157.9	6.12	1.00	32.54	0.00	9.67	227.8	4.24
ABF-Pre7-4C-W	1.00	1322.81	0.00	14.28	119.9	11.91	1.00	354.47	0.00	14.29	292.1	4.89	1.00	105.86	0.00	14.29	488.7	2.92	1.00	32.54	0.00	14.29	738.5	1.93
AMG-4C-W	1.03	1062.24	0.01	4.88	67.2	7.26	1.03	287.19	0.01	4.88	106.7	4.57	1.03	81.74	0.01	4.88	129.8	3.76	1.03	26.21	0.01	4.88	144.6	3.37

Original condition number =786.6

(b) theoretical convergence results

Table 106: Poisson blend, 33×33 image

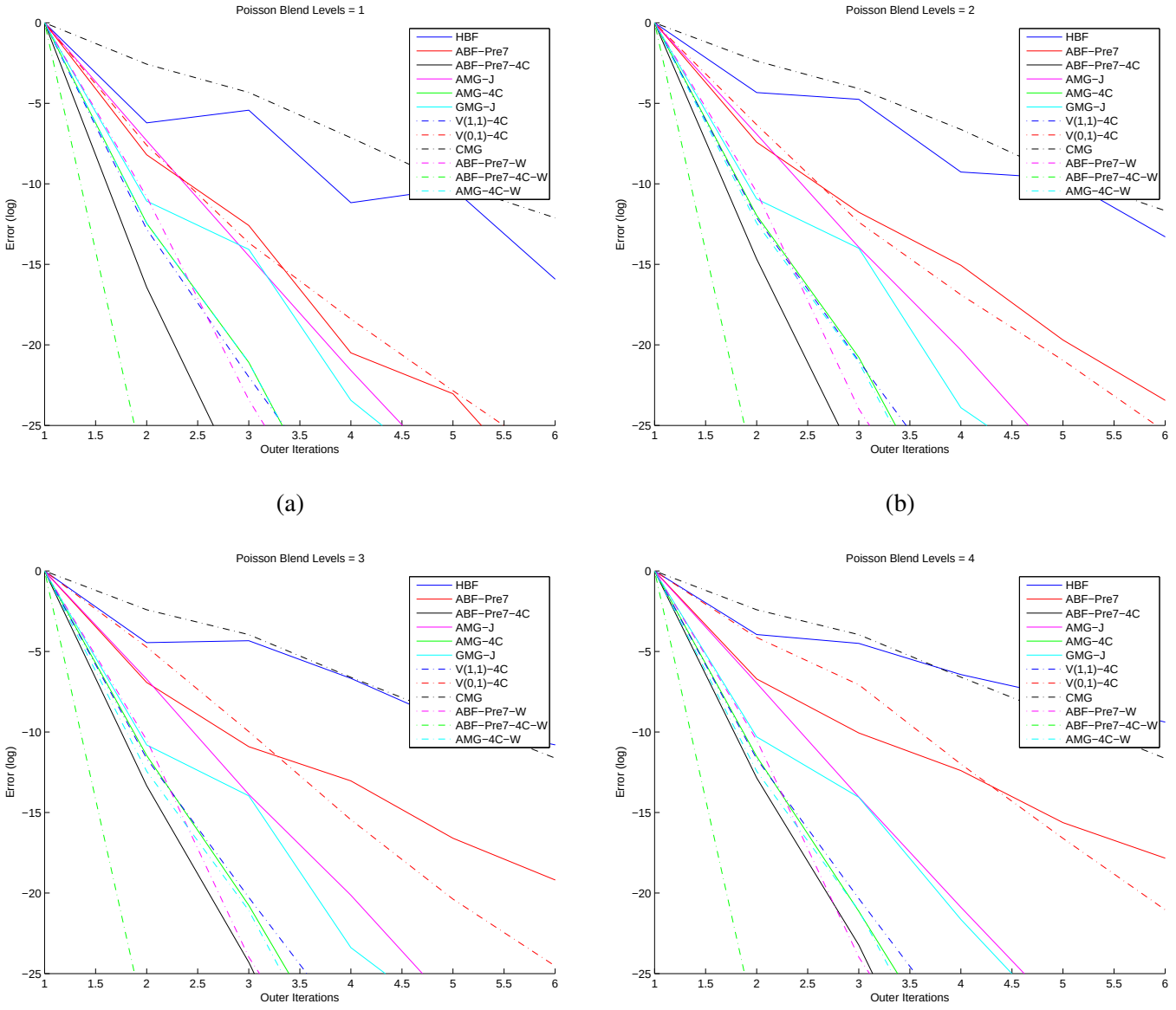


Figure 116: Log error vs. its, Poisson blend, 33×33 image

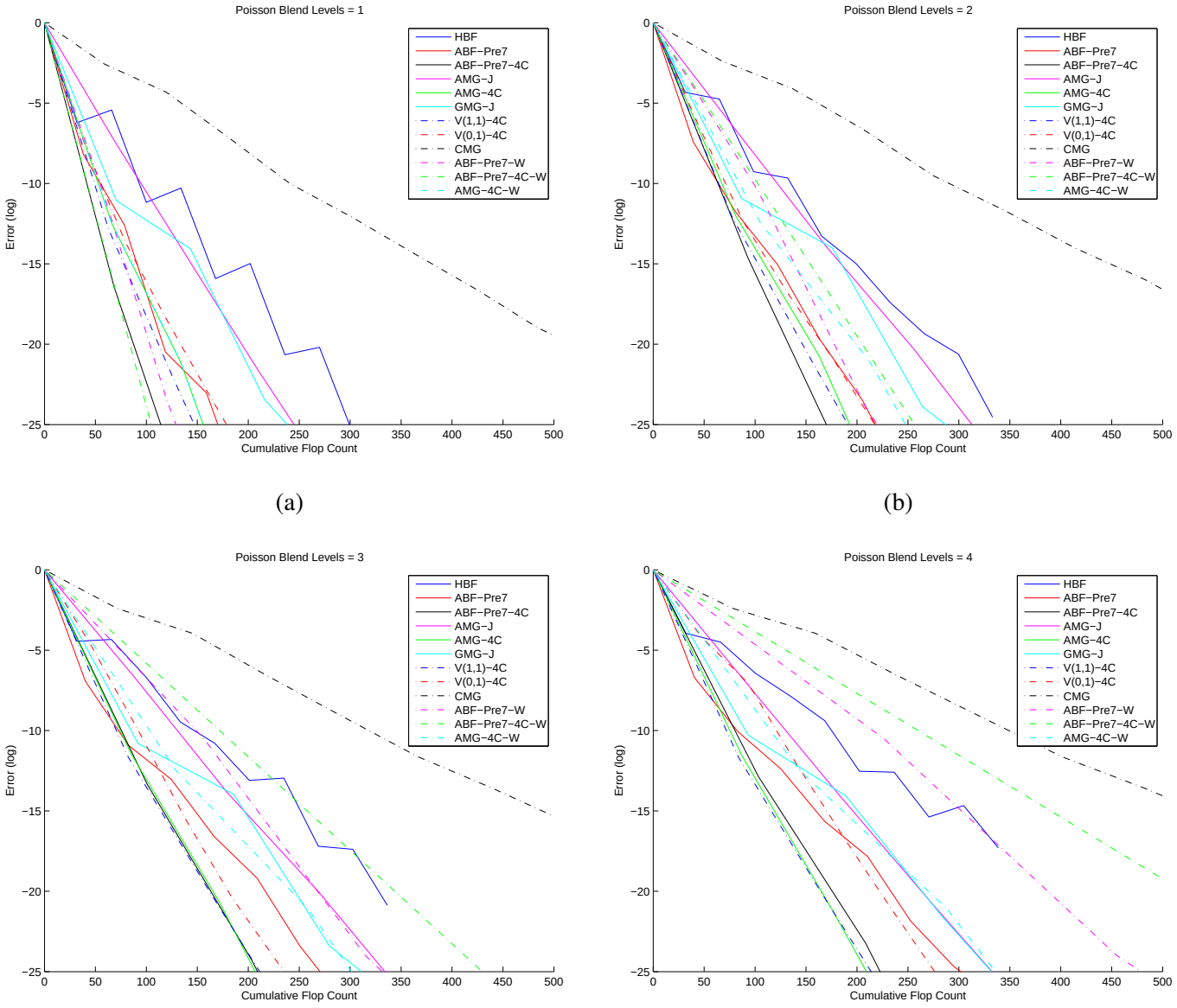
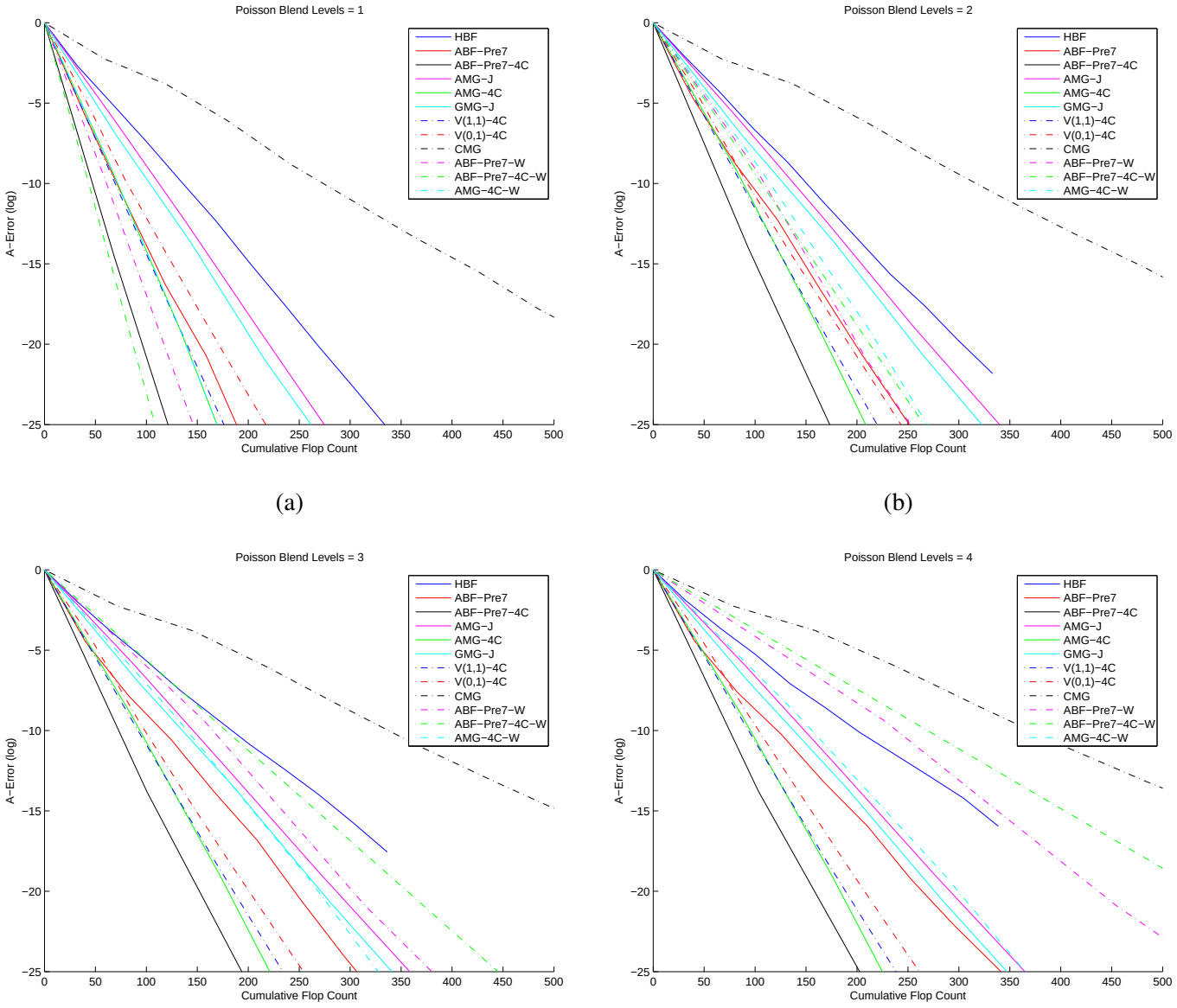


Figure 117: Log error vs. flops, Poisson blend, 33×33 image



(a) (b) (c) (d)

Figure 118: Log A-error vs. flops, Poisson blend, 33×33 image

	$L=1$				$L=2$				$L=3$				$L=4$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.16	1.83	33.8	5.41	0.16	1.81	33.3	5.44	0.22	1.49	33.6	4.43	0.26	1.33	33.9	3.93
ABF-Pre7	0.02	3.97	40.0	9.93	0.04	3.17	41.2	7.71	0.08	2.56	41.9	6.11	0.10	2.29	42.3	5.41
ABF-Pre7-4C	0.00	10.49	69.6	15.06	0.00	9.35	94.7	9.88	0.00	8.74	102.1	8.56	0.00	8.56	104.6	8.19
AMG-J	0.01	4.90	70.1	7.00	0.01	4.73	85.6	5.52	0.01	4.96	90.5	5.48	0.01	4.51	92.1	4.90
AMG-4C	0.00	7.93	67.2	11.79	0.00	7.62	82.1	9.28	0.00	7.61	86.7	8.78	0.00	7.60	88.3	8.61
GMG-J	0.00	5.36	72.3	7.41	0.00	5.49	88.4	6.21	0.00	5.33	93.4	5.70	0.01	4.93	95.1	5.18
V(1,1)-4C	0.00	7.21	62.7	11.50	0.00	6.98	77.6	8.99	0.00	7.06	82.2	8.59	0.00	7.03	83.8	8.39
V(0,1)-4C	0.02	3.70	39.7	9.32	0.03	3.60	44.2	8.14	0.03	3.40	46.0	7.40	0.05	3.09	46.6	6.63
CMG	0.16	1.81	60.6	2.98	0.18	1.69	69.1	2.45	0.19	1.68	73.5	2.28	0.19	1.67	80.5	2.08
ABF-Pre7-W	0.00	7.90	60.3	13.10	0.00	7.99	104.9	7.62	0.00	7.99	157.9	5.06	0.00	7.99	227.8	3.51
ABF-Pre7-4C-W	0.00	18.75	119.9	15.64	0.00	18.75	292.1	6.42	0.00	18.75	488.7	3.84	0.00	18.75	738.5	2.54
AMG-4C-W	0.00	7.93	67.2	11.79	0.00	7.93	106.7	7.43	0.00	7.93	129.8	6.11	0.00	7.93	144.6	5.48

(a) empirical convergence results

	$L=1$						$L=2$						$L=3$						$L=4$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	1.71	1061.85	0.13	2.02	33.8	5.97	1.95	287.05	0.17	1.80	33.3	5.40	2.87	81.54	0.26	1.36	33.6	4.03	5.06	25.85	0.38	0.96	33.9	2.82
ABF-Pre7	1.31	1322.81	0.07	2.69	40.0	6.73	1.31	354.47	0.07	2.68	41.2	6.52	1.38	105.86	0.08	2.53	41.9	6.04	1.63	32.54	0.12	2.11	42.3	4.98
ABF-Pre7-4C	1.01	1322.81	0.00	5.82	69.6	8.36	1.02	354.47	0.01	5.25	94.7	5.54	1.01	105.86	0.00	5.69	102.1	5.57	1.01	32.54	0.00	5.95	104.6	5.69
AMG-J	1.00	1062.24	0.00	7.33	70.1	10.46	1.01	287.19	0.00	6.23	85.6	7.28	1.04	81.74	0.01	4.62	90.5	5.11	1.12	26.21	0.03	3.53	92.1	3.83
AMG-4C	1.03	1062.24	0.01	4.88	67.2	7.26	1.04	287.19	0.01	4.75	82.1	5.79	1.05	81.74	0.01	4.45	86.7	5.13	1.07	26.21	0.02	4.12	88.3	4.67
GMG-J	1.19	1061.85	0.04	3.14	72.3	4.35	1.18	287.05	0.04	3.18	88.4	3.60	1.17	81.54	0.04	3.26	93.4	3.49	1.14	25.85	0.03	3.43	95.1	3.61
V(1,1)-4C	1.03	1062.24	0.01	4.88	62.7	7.78	1.04	287.19	0.01	4.75	77.6	6.13	1.05	81.74	0.01	4.45	82.2	5.41	1.07	26.21	0.02	4.12	83.8	4.92
V(0,1)-4C	1.12	1062.24	0.03	3.57	39.7	8.99	1.15	287.19	0.04	3.35	44.2	7.56	1.17	81.74	0.04	3.26	46.0	7.09	1.18	26.21	0.04	3.16	46.6	6.78
CMG	5.89	4953.66	0.42	0.88	60.6	1.45	5.88	1875.04	0.42	0.88	69.1	1.27	6.59	1032.83	0.44	0.82	73.5	1.12	6.59	250.46	0.44	0.82	80.5	1.02
ABF-Pre7-W	1.00	1322.81	0.00	9.70	60.3	16.08	1.00	354.47	0.00	9.67	104.9	9.21	1.00	105.86	0.00	9.67	157.9	6.12	1.00	32.54	0.00	9.67	227.8	4.24
ABF-Pre7-4C-W	1.00	1322.81	0.00	14.28	119.9	11.91	1.00	354.47	0.00	14.29	292.1	4.89	1.00	105.86	0.00	14.29	488.7	2.92	1.00	32.54	0.00	14.29	738.5	1.93
AMG-4C-W	1.03	1062.24	0.01	4.88	67.2	7.26	1.03	287.19	0.01	4.88	106.7	4.57	1.03	81.74	0.01	4.88	129.8	3.76	1.03	26.21	0.01	4.88	144.6	3.37

Original condition number =786.6

(b) theoretical convergence results

Table 107: Poisson blend, 33×33 image

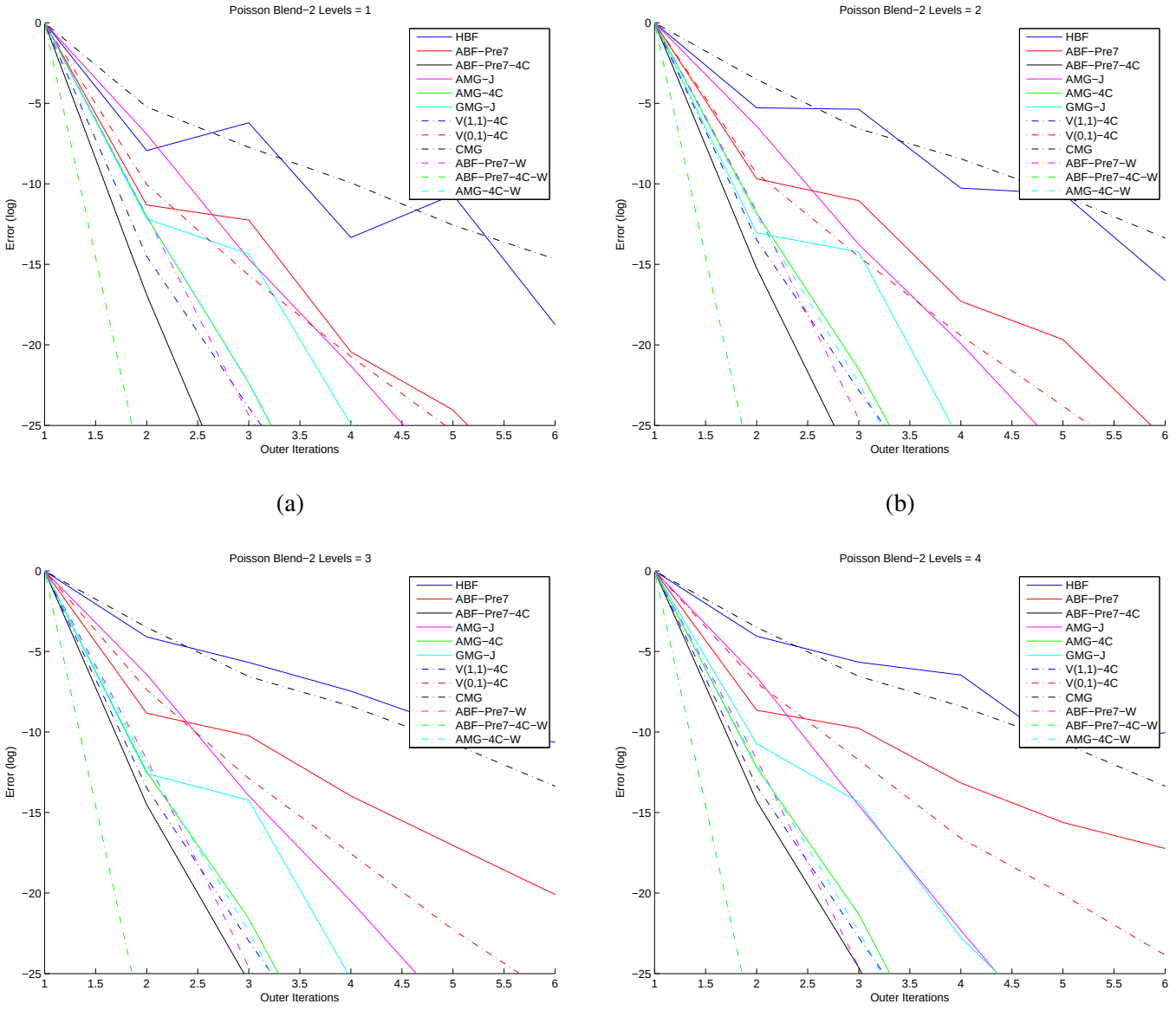


Figure 119: Log error vs. its, Poisson blend, 33×33 image

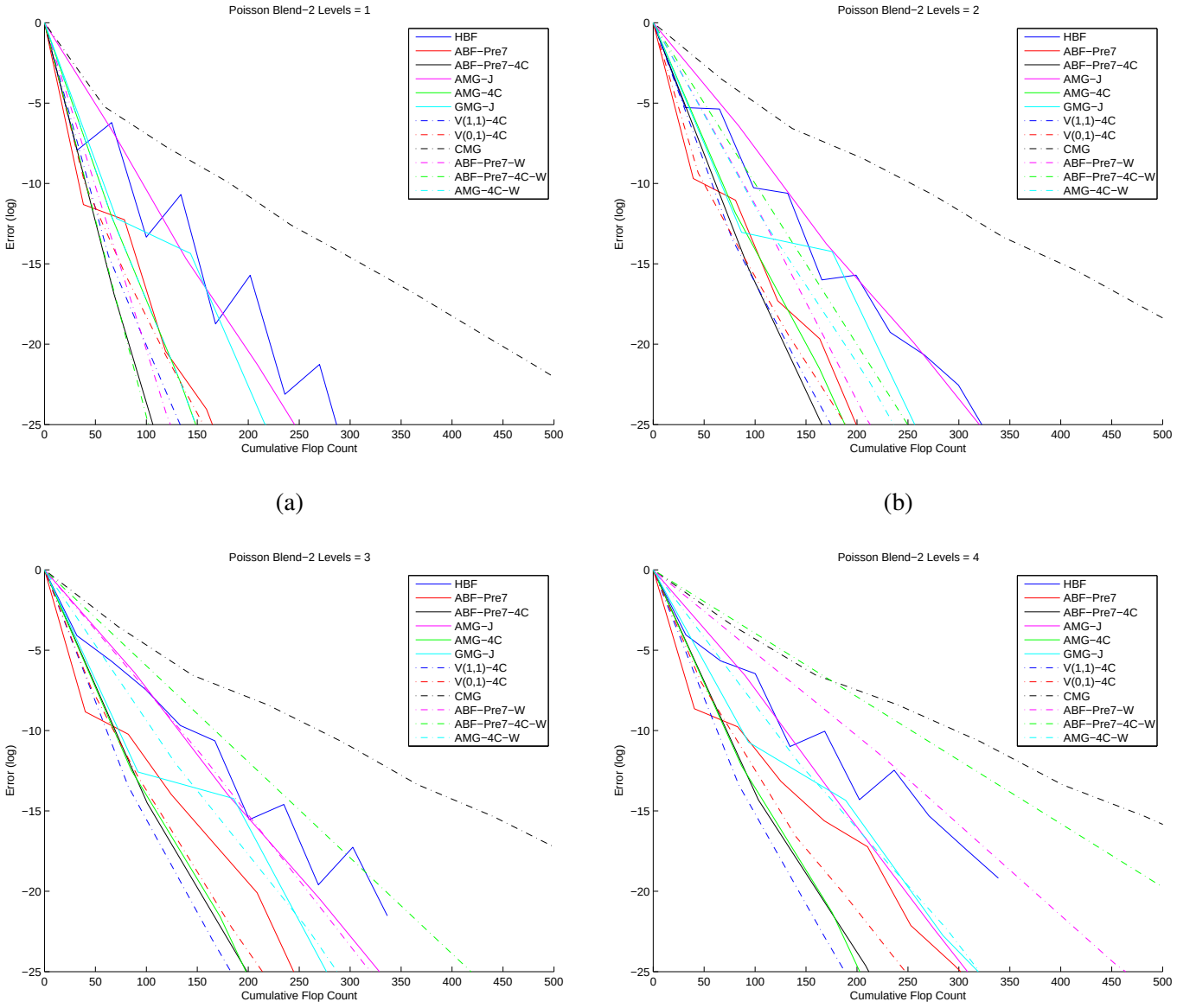


Figure 120: Log error vs. flops, Poisson blend, 33×33 image

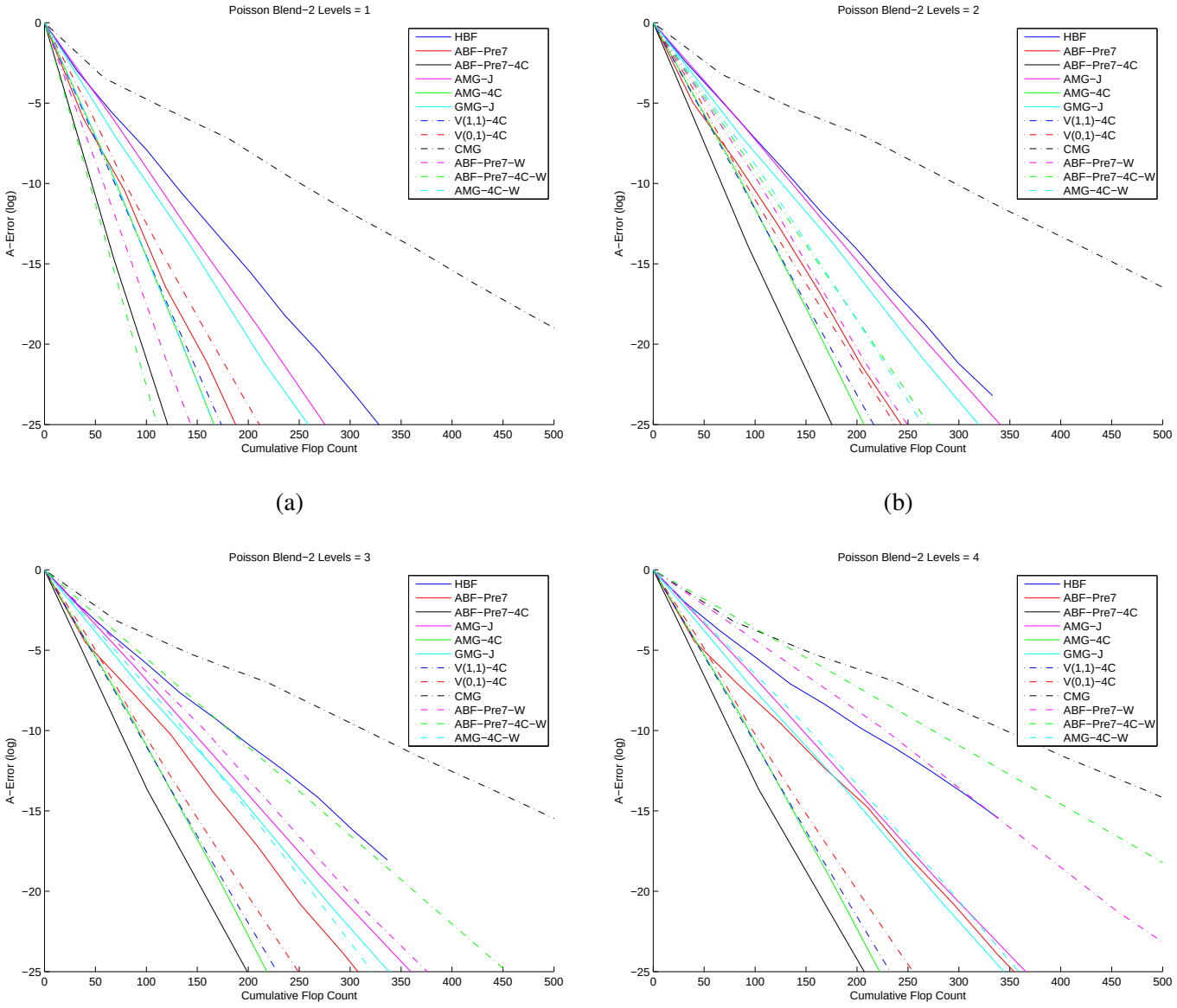


Figure 121: Log A-error vs. flops, Poisson blend, 33×33 image

	$L=3$				$L=4$				$L=5$				$L=6$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.30	1.19	32.5	3.67	0.36	1.02	32.4	3.15	0.39	0.95	32.5	2.93	0.40	0.91	32.5	2.80
ABF-Pre7	0.09	2.36	41.0	5.75	0.13	2.06	41.1	5.03	0.16	1.84	41.1	4.47	0.16	1.84	41.1	4.48
ABF-Pre7-4C	0.00	8.08	98.7	8.18	0.00	7.92	100.1	7.91	0.00	7.83	100.5	7.79	0.00	7.82	100.6	7.77
AMG-J	0.02	4.15	87.7	4.74	0.02	3.94	88.6	4.45	0.02	3.82	88.8	4.30	0.02	3.91	88.9	4.40
AMG-4C	0.00	6.28	84.0	7.47	0.00	5.89	84.9	6.93	0.00	5.77	85.2	6.77	0.00	5.73	85.2	6.73
GMG-J	0.01	4.64	90.6	5.12	0.01	4.42	91.5	4.83	0.01	4.30	91.8	4.69	0.02	4.05	91.9	4.41
V(1,1)-4C	0.01	4.61	79.5	5.80	0.02	4.15	80.3	5.17	0.02	4.06	80.6	5.04	0.02	4.06	80.6	5.03
V(0,1)-4C	0.09	2.36	44.3	5.33	0.14	1.95	44.5	4.38	0.17	1.80	44.6	4.03	0.17	1.77	44.7	3.96
CMG	0.26	1.36	67.8	2.01	0.26	1.36	70.1	1.94	0.26	1.36	71.1	1.91	0.26	1.36	72.3	1.87
ABF-Pre7-W	0.00	7.68	145.6	5.28	0.00	7.68	185.4	4.14	0.00	7.68	226.5	3.39	0.00	7.68	268.7	2.86
ABF-Pre7-4C-W	0.00	16.71	445.1	3.75	0.00	16.71	604.9	2.76	0.00	16.71	766.0	2.18	0.00	16.71	928.2	1.80
AMG-4C-W	0.00	7.28	122.3	5.95	0.00	7.28	131.4	5.54	0.00	7.28	136.2	5.35	0.00	7.28	138.6	5.25

(a) empirical convergence results

	$L=3$						$L=4$						$L=5$						$L=6$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	3.11	87.51	0.28	1.29	32.5	3.96	4.87	24.16	0.38	0.98	32.4	3.01	6.88	9.00	0.45	0.80	32.5	2.47	8.95	8.58	0.50	0.70	32.5	2.14
ABF-Pre7	1.38	81.03	0.08	2.53	41.0	6.17	1.41	22.74	0.09	2.46	41.1	6.00	1.49	10.62	0.10	2.31	41.1	5.63	1.56	7.80	0.11	2.20	41.1	5.35
ABF-Pre7-4C	1.08	81.03	0.02	3.97	98.7	4.02	1.07	22.74	0.02	4.08	100.1	4.08	1.06	10.62	0.01	4.20	100.5	4.18	1.06	7.80	0.01	4.21	100.6	4.18
AMG-J	1.03	87.46	0.01	4.79	87.7	5.47	1.13	24.07	0.03	3.48	88.6	3.93	1.28	8.62	0.06	2.80	88.8	3.15	1.32	5.53	0.07	2.66	88.9	2.99
AMG-4C	1.02	87.46	0.00	5.43	84.0	6.46	1.03	24.07	0.01	4.81	84.9	5.66	1.05	8.62	0.01	4.35	85.2	5.10	1.06	5.53	0.01	4.22	85.2	4.95
GMG-J	1.21	87.51	0.05	3.06	90.6	3.37	1.21	24.16	0.05	3.06	91.5	3.35	1.20	9.00	0.05	3.08	91.8	3.35	1.20	8.58	0.04	3.10	91.9	3.38
V(1,1)-4C	1.02	87.46	0.00	5.43	79.5	6.83	1.03	24.07	0.01	4.81	80.3	5.98	1.05	8.62	0.01	4.35	80.6	5.39	1.06	5.53	0.01	4.22	80.6	5.23
V(0,1)-4C	1.24	87.46	0.05	2.91	44.3	6.57	1.49	24.07	0.10	2.31	44.5	5.19	1.69	8.62	0.13	2.04	44.6	4.57	1.71	5.53	0.13	2.02	44.7	4.51
CMG	3.86	1292.17	0.33	1.12	67.8	1.66	3.86	504.92	0.33	1.12	70.1	1.60	4.16	268.76	0.34	1.07	71.1	1.51	4.16	88.12	0.34	1.07	72.3	1.48
ABF-Pre7-W	-	-	-	-	145.6	-	-	-	-	-	185.4	-	-	-	-	-	226.5	-	-	-	-	-	268.7	-
ABF-Pre7-4C-W	-	-	-	-	445.1	-	-	-	-	-	604.9	-	-	-	-	-	766.0	-	-	-	-	-	928.2	-
AMG-4C-W	-	-	-	-	122.3	-	-	-	-	-	131.4	-	-	-	-	-	136.2	-	-	-	-	-	138.6	-

Original condition number =4770.8

(b) theoretical convergence results

Table 108: Poisson blend, 128×128 image

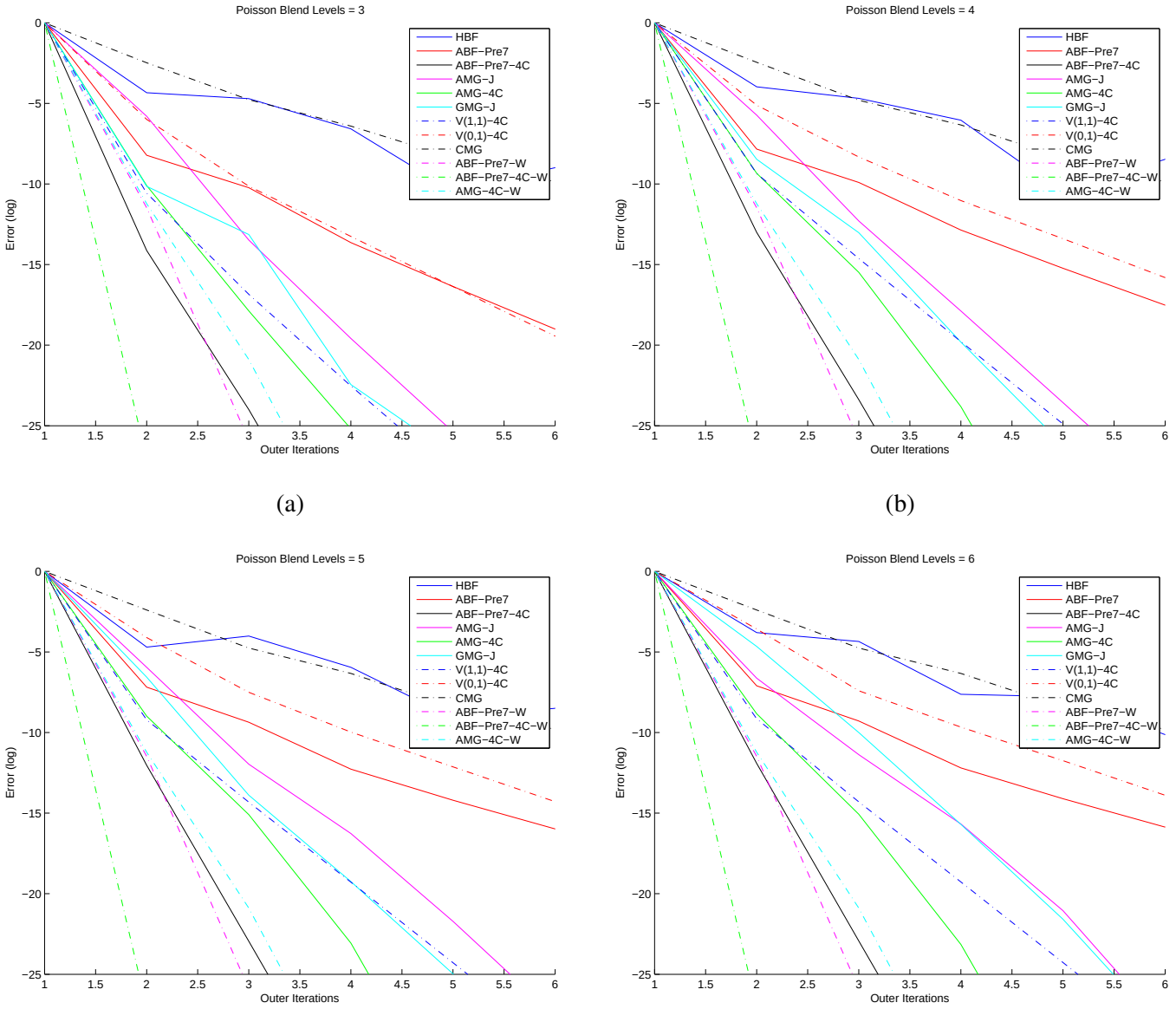


Figure 122: Log error vs. its, Poisson blend, 128×128 image

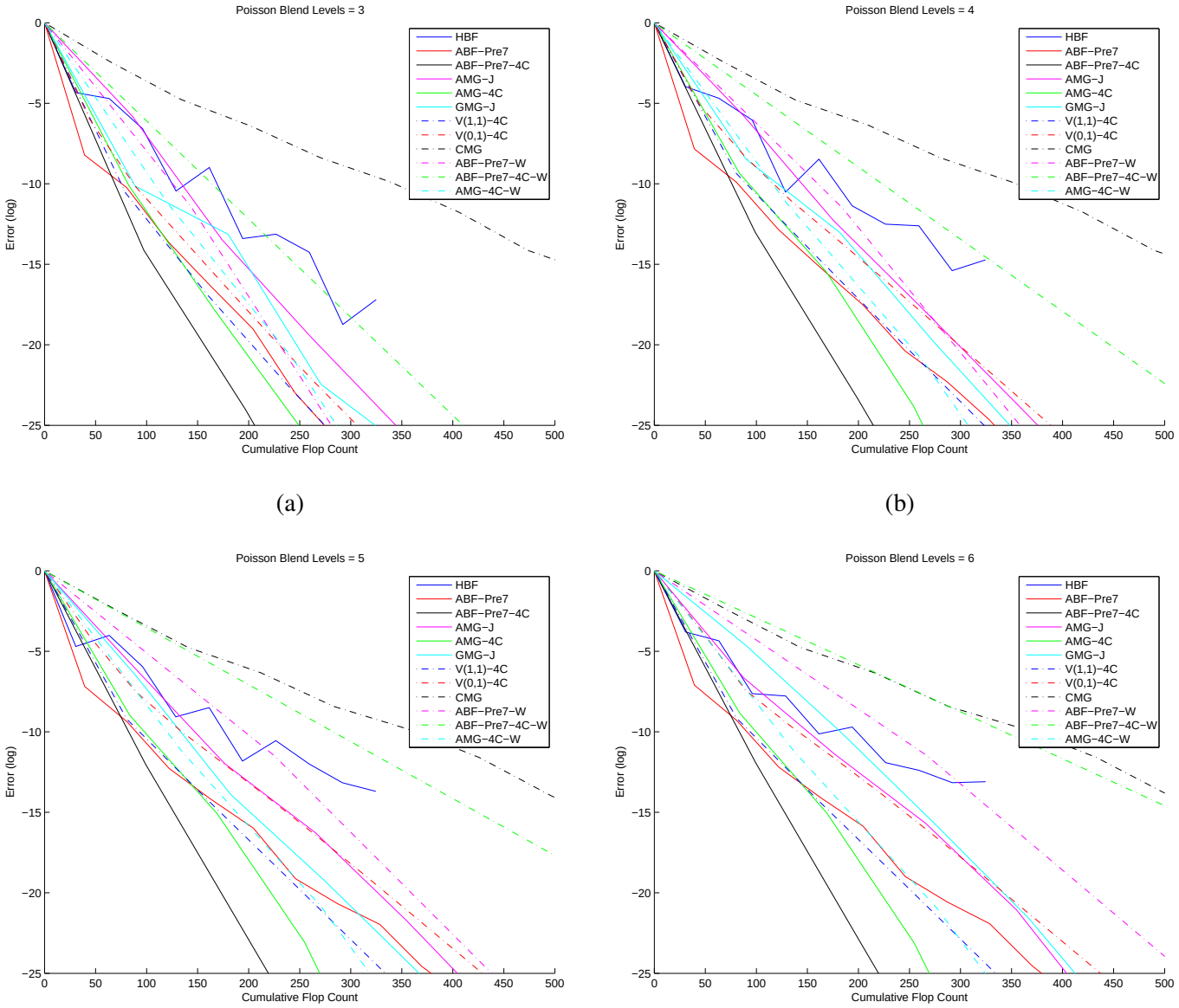


Figure 123: Log error vs. flops, Poisson blend, 128×128 image

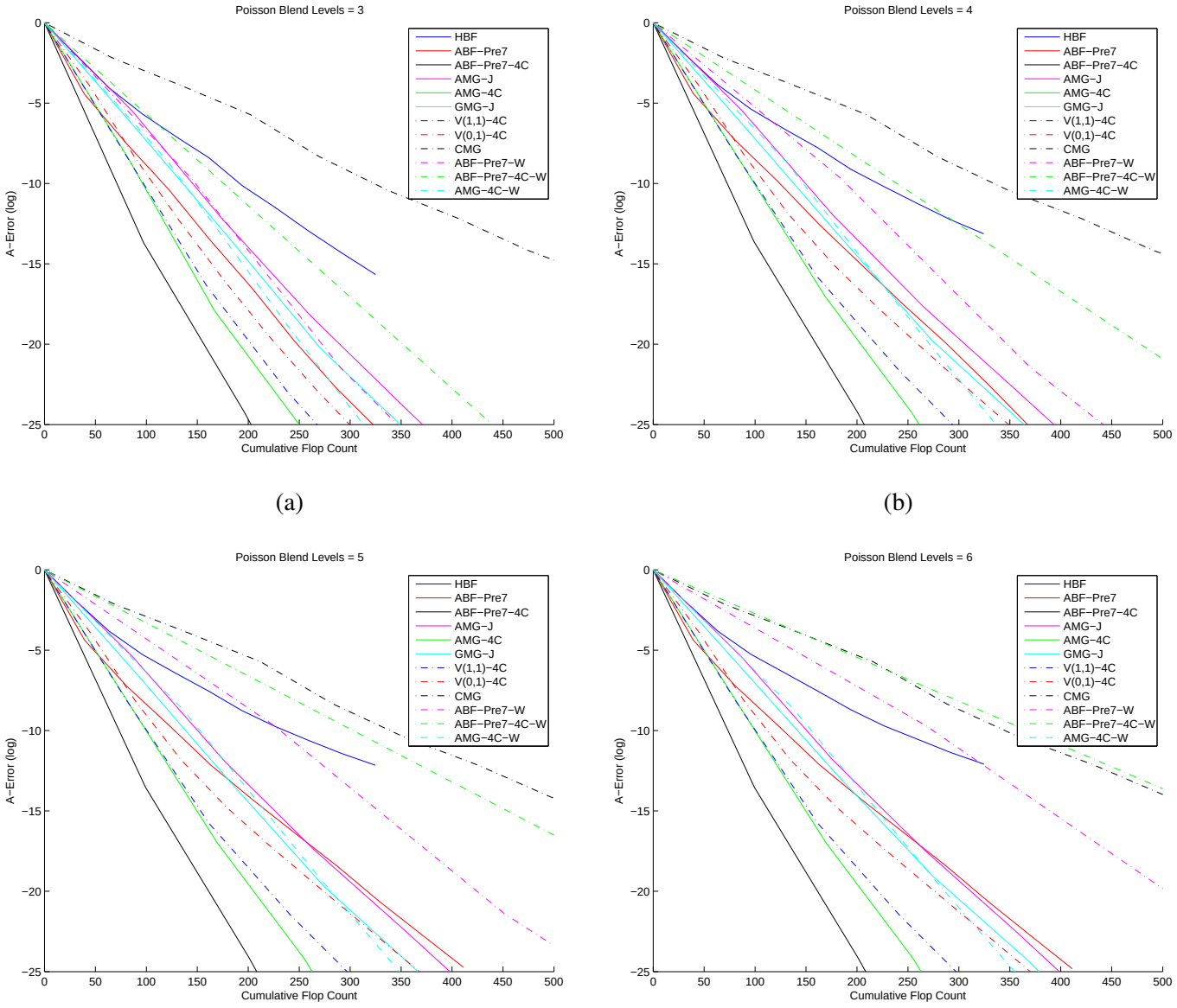


Figure 124: Log A-error vs. flops, Poisson blend, 128×128 image

	$L=4$				$L=5$				$L=6$				$L=7$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.30	1.19	32.5	3.66	0.34	1.07	32.5	3.29	0.37	0.99	32.5	3.04	0.46	0.78	32.5	2.41
ABF-Pre7	0.11	2.25	41.1	5.46	0.13	2.07	41.1	5.03	0.13	2.00	41.1	4.87	0.14	1.99	41.1	4.83
ABF-Pre7-4C	0.00	8.95	100.2	8.93	0.00	8.72	100.5	8.68	0.00	8.55	100.6	8.50	0.00	8.55	100.7	8.49
AMG-J	0.01	4.22	88.6	4.77	0.02	4.09	88.8	4.60	0.02	4.08	88.9	4.59	0.02	3.96	88.9	4.46
AMG-4C	0.00	6.24	84.9	7.34	0.00	6.13	85.1	7.20	0.00	6.13	85.1	7.20	0.00	6.14	85.2	7.21
GMG-J	0.01	4.89	91.6	5.34	0.01	4.47	91.8	4.87	0.02	4.13	91.9	4.50	0.02	4.05	91.9	4.41
V(1,1)-4C	0.01	5.12	80.4	6.37	0.01	5.11	80.6	6.35	0.01	5.11	80.6	6.34	0.01	5.12	80.7	6.34
V(0,1)-4C	0.08	2.49	44.6	5.57	0.09	2.40	44.6	5.38	0.09	2.39	44.7	5.36	0.09	2.40	44.7	5.36
CMG	0.20	1.61	73.3	2.19	0.20	1.60	72.6	2.20	0.20	1.60	73.1	2.19	0.20	1.60	73.4	2.18
ABF-Pre7-W	0.00	7.58	193.0	3.93	0.00	7.58	232.2	3.27	0.00	7.58	272.1	2.79	0.00	7.58	313.2	2.42
ABF-Pre7-4C-W	0.00	18.96	612.5	3.10	0.00	18.96	771.7	2.46	0.00	18.96	931.6	2.04	0.00	18.96	1092.7	1.74
AMG-4C-W	0.00	7.25	131.9	5.50	0.00	7.25	136.3	5.32	0.00	7.25	138.6	5.23	0.00	7.25	139.8	5.19

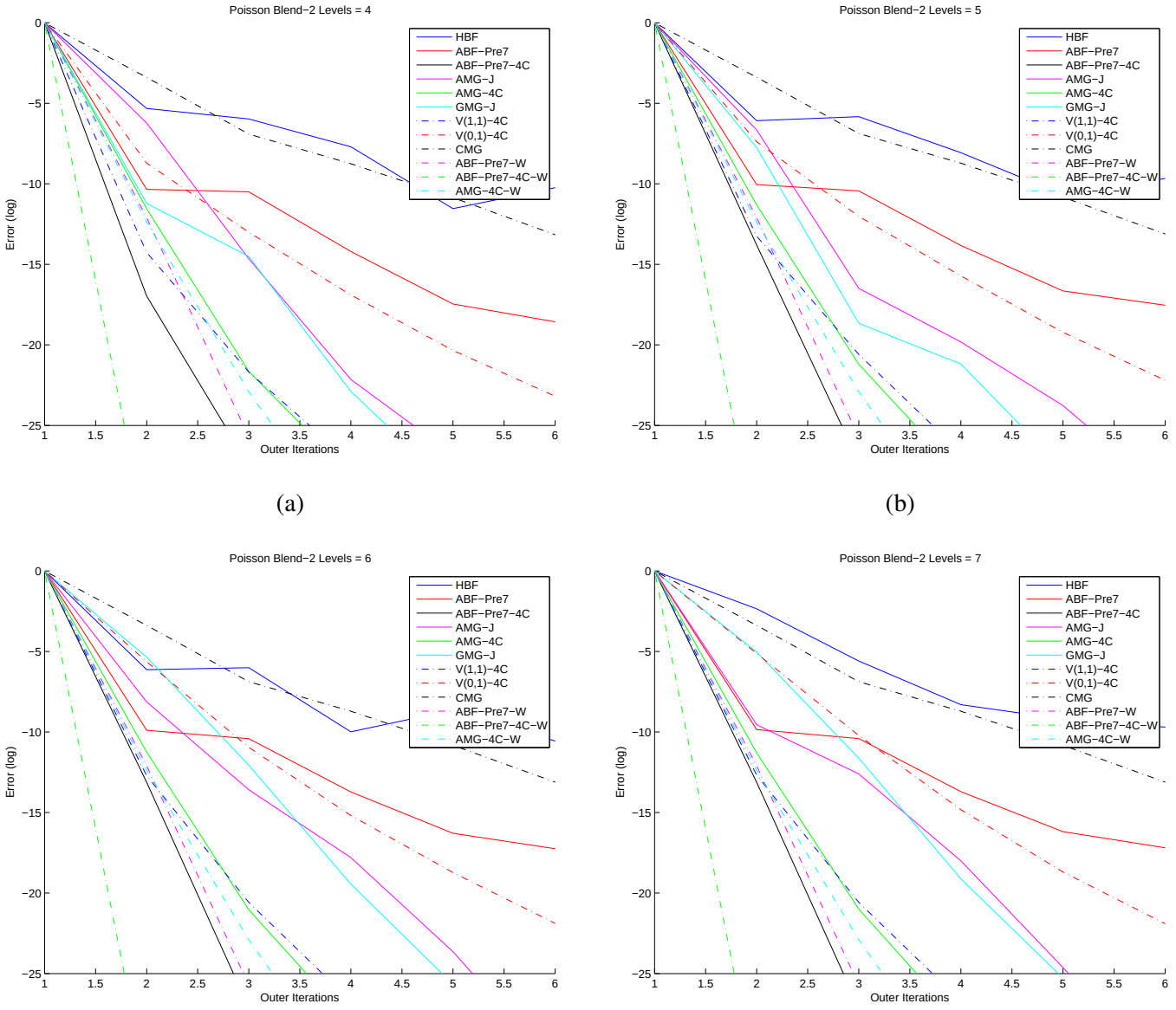
(a) empirical convergence results

	$L=4$						$L=5$						$L=6$						$L=7$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	3.50	24.89	0.30	1.19	32.5	3.67	6.27	9.49	0.43	0.85	32.5	2.60	14.81	8.73	0.59	0.53	32.5	1.64	23.64	15.72	0.66	0.42	32.5	1.29
ABF-Pre7	1.38	23.05	0.08	2.52	41.1	6.13	1.38	11.10	0.08	2.52	41.1	6.13	1.38	8.89	0.08	2.51	41.1	6.11	1.39	7.83	0.08	2.51	41.1	6.10
ABF-Pre7-4C	1.08	23.05	0.02	3.90	100.2	3.89	1.08	11.10	0.02	4.00	100.5	3.98	1.07	8.89	0.02	4.03	100.6	4.01	1.07	7.83	0.02	4.03	100.7	4.01
AMG-J	1.01	24.77	0.00	6.58	88.6	7.42	1.02	9.16	0.01	5.25	88.8	5.91	1.09	5.70	0.02	3.79	88.9	4.26	1.12	4.15	0.03	3.54	88.9	3.99
AMG-4C	1.00	24.77	0.00	6.73	84.9	7.93	1.01	9.16	0.00	6.52	85.1	7.66	1.01	5.70	0.00	5.90	85.1	6.93	1.01	4.15	0.00	5.85	85.2	6.87
GMG-J	1.17	24.89	0.04	3.21	91.6	3.51	1.11	9.49	0.03	3.67	91.8	4.00	1.02	8.73	0.00	5.41	91.9	5.89	1.01	15.72	0.00	5.62	91.9	6.12
V(1,1)-4C	1.00	24.77	0.00	6.73	80.4	8.37	1.01	9.16	0.00	6.52	80.6	8.09	1.01	5.70	0.00	5.90	80.6	7.31	1.01	4.15	0.00	5.85	80.7	7.25
V(0,1)-4C	1.20	24.77	0.05	3.10	44.6	6.94	1.26	9.16	0.06	2.85	44.6	6.38	1.36	5.70	0.08	2.56	44.7	5.73	1.40	4.15	0.08	2.48	44.7	5.55
CMG	1.36	505.48	0.08	2.56	73.3	3.49	1.38	310.44	0.08	2.52	72.6	3.47	1.38	127.02	0.08	2.52	73.1	3.45	1.38	74.77	0.08	2.51	73.4	3.42
ABF-Pre7-W	-	-	-	-	193.0	-	-	-	-	-	232.2	-	-	-	-	-	272.1	-	-	-	-	-	313.2	-
ABF-Pre7-4C-W	-	-	-	-	612.5	-	-	-	-	-	771.7	-	-	-	-	-	931.6	-	-	-	-	-	1092.7	-
AMG-4C-W	-	-	-	-	131.9	-	-	-	-	-	136.3	-	-	-	-	-	138.6	-	-	-	-	-	139.8	-

Original condition number =7640.5

(b) theoretical convergence results

Table 109: Poisson blend, 512×512 image



(a) (b)

(c) (d)

Figure 125: Log error vs. its, Poisson blend, 512×512 image

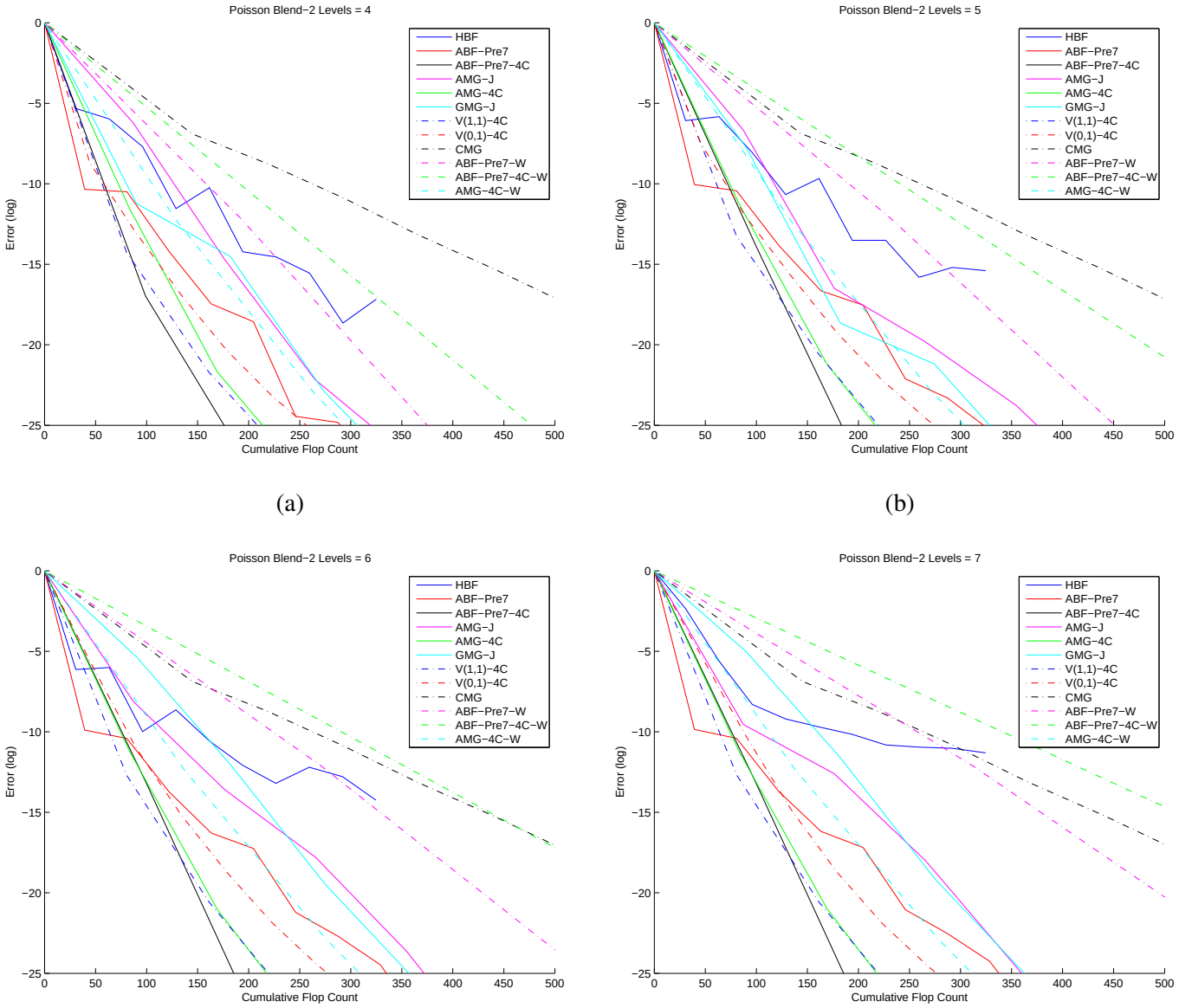


Figure 126: Log error vs. flops, Poisson blend, 512×512 image

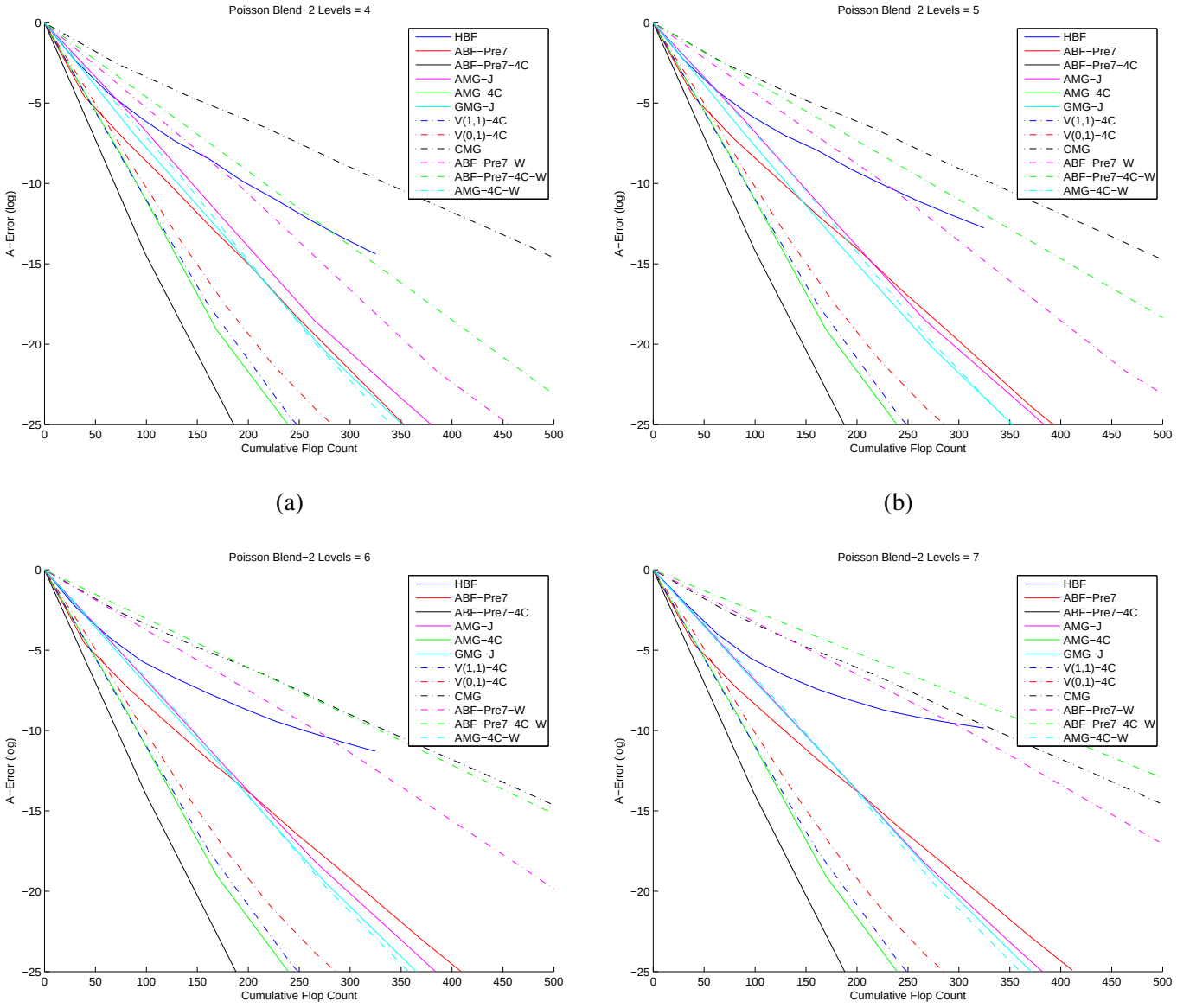


Figure 127: Log A-error vs. flops, Poisson blend, 512×512 image

	$L=7$				$L=8$				$L=9$				$L=10$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.40	0.90	32.5	2.79	0.44	0.81	32.5	2.51	0.44	0.82	32.5	2.52	0.43	0.84	32.5	2.58
ABF-Pre7	0.10	2.27	41.1	5.52	0.10	2.27	41.1	5.52	0.10	2.27	41.1	5.52	0.10	2.27	41.1	5.52
ABF-Pre7-4C	0.00	9.58	100.7	9.52	0.00	9.58	100.7	9.52	0.00	9.58	100.7	9.52	0.00	9.58	100.7	9.52
AMG-J	0.01	4.58	88.9	5.15	0.01	4.58	88.9	5.15	0.01	4.58	88.9	5.15	0.01	4.58	88.9	5.15
AMG-4C	0.00	6.56	85.3	7.69	0.00	6.56	85.3	7.69	0.00	6.56	85.3	7.69	0.00	6.56	85.3	7.69
GMG-J	0.01	4.62	91.9	5.03	0.01	4.60	91.9	5.00	0.01	4.59	91.9	4.99	0.01	4.58	91.9	4.98
V(1,1)-4C	0.00	5.42	80.7	6.72	0.00	5.42	80.7	6.72	0.00	5.42	80.7	6.72	0.00	5.42	80.7	6.72
V(0,1)-4C	0.09	2.45	44.7	5.48	0.09	2.45	44.7	5.48	0.09	2.45	44.7	5.48	0.09	2.45	44.7	5.48
CMG	0.18	1.70	73.8	2.31	0.18	1.70	74.1	2.30	0.18	1.70	74.2	2.29	0.18	1.70	74.4	2.29

(a) empirical convergence results

Table 110: Poisson blend, 2048×2048 image

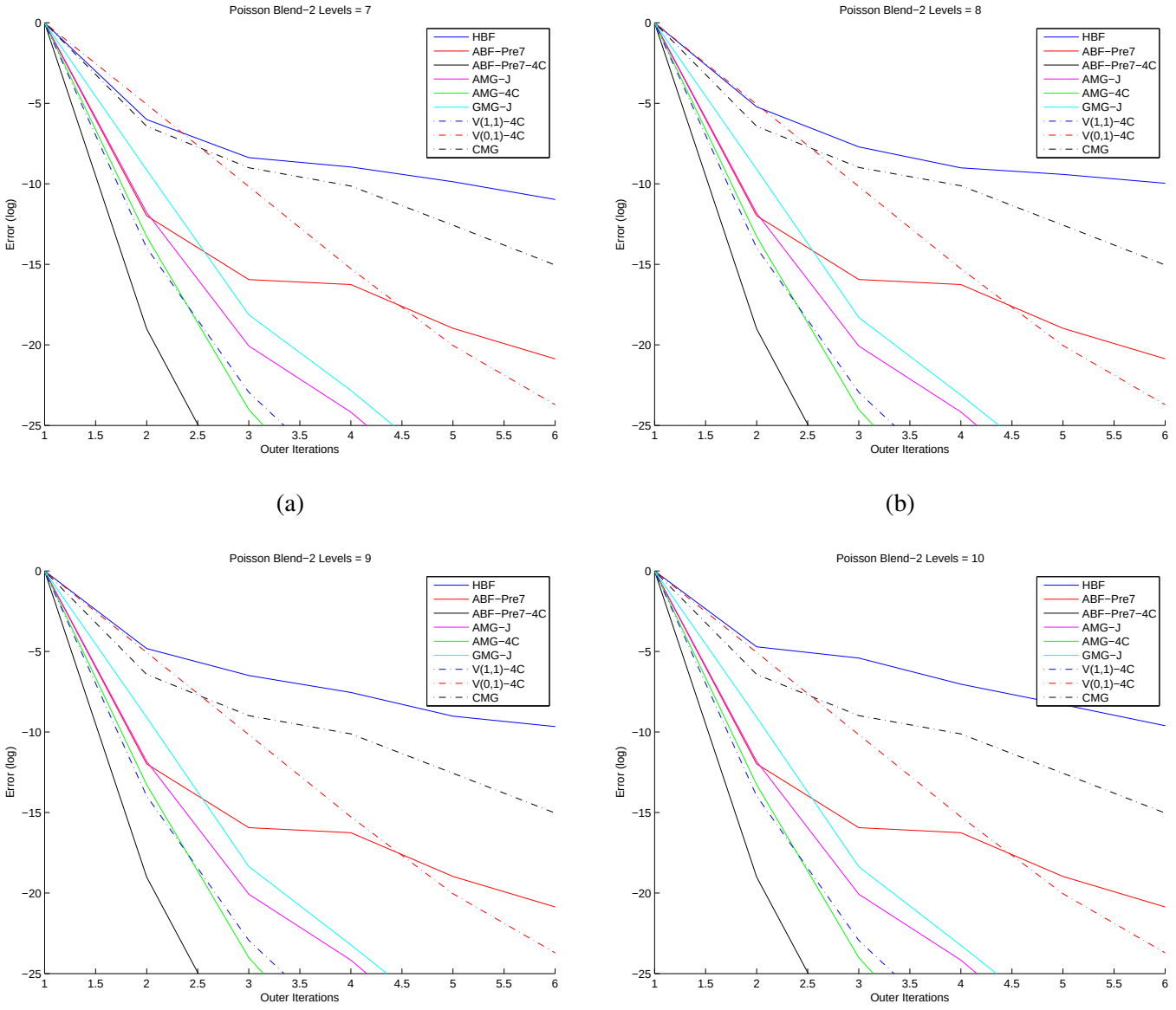


Figure 128: Log error vs. its, Poisson blend, 2048×2048 image

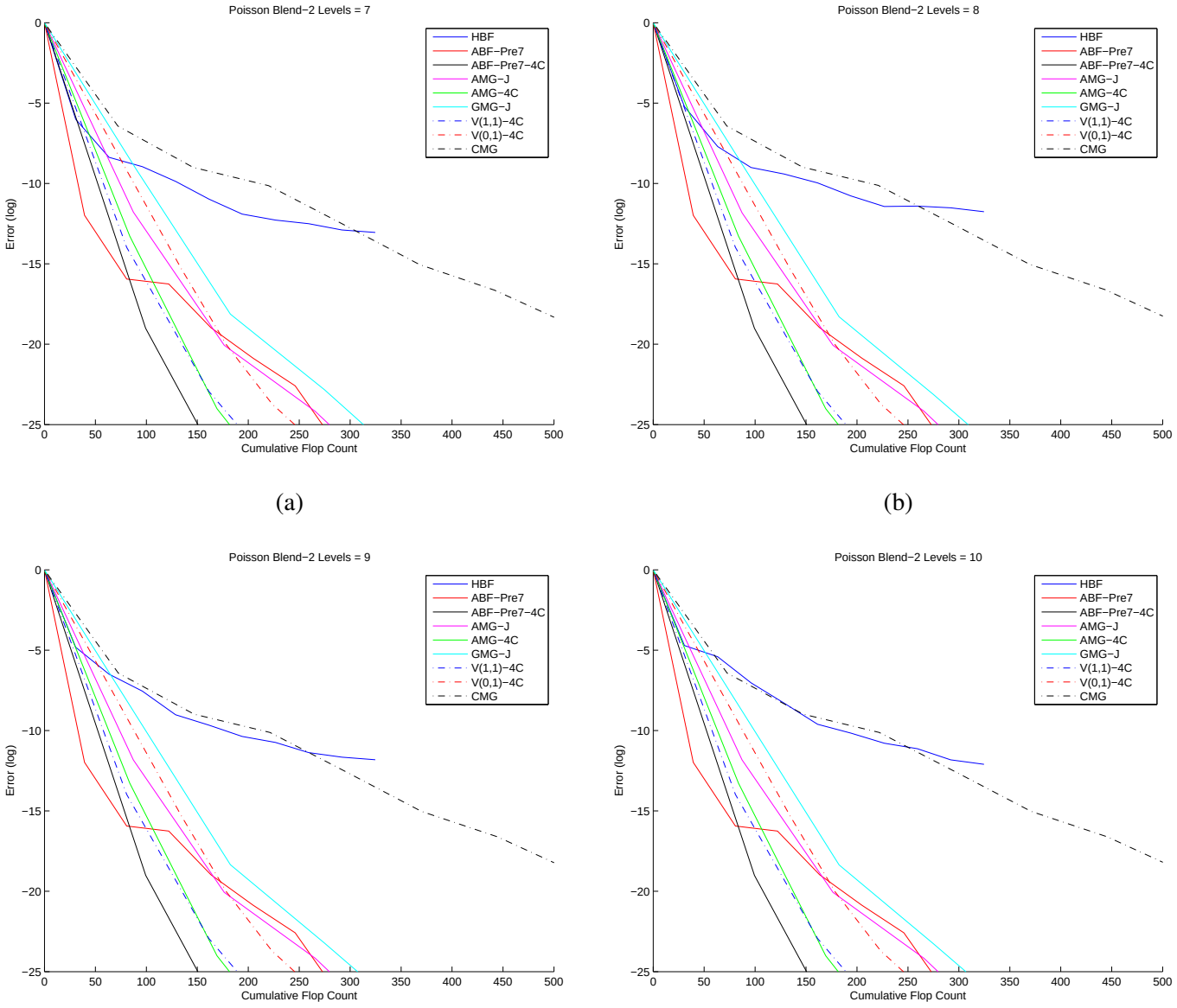


Figure 129: Log error vs. flops, Poisson blend, 2048×2048 image

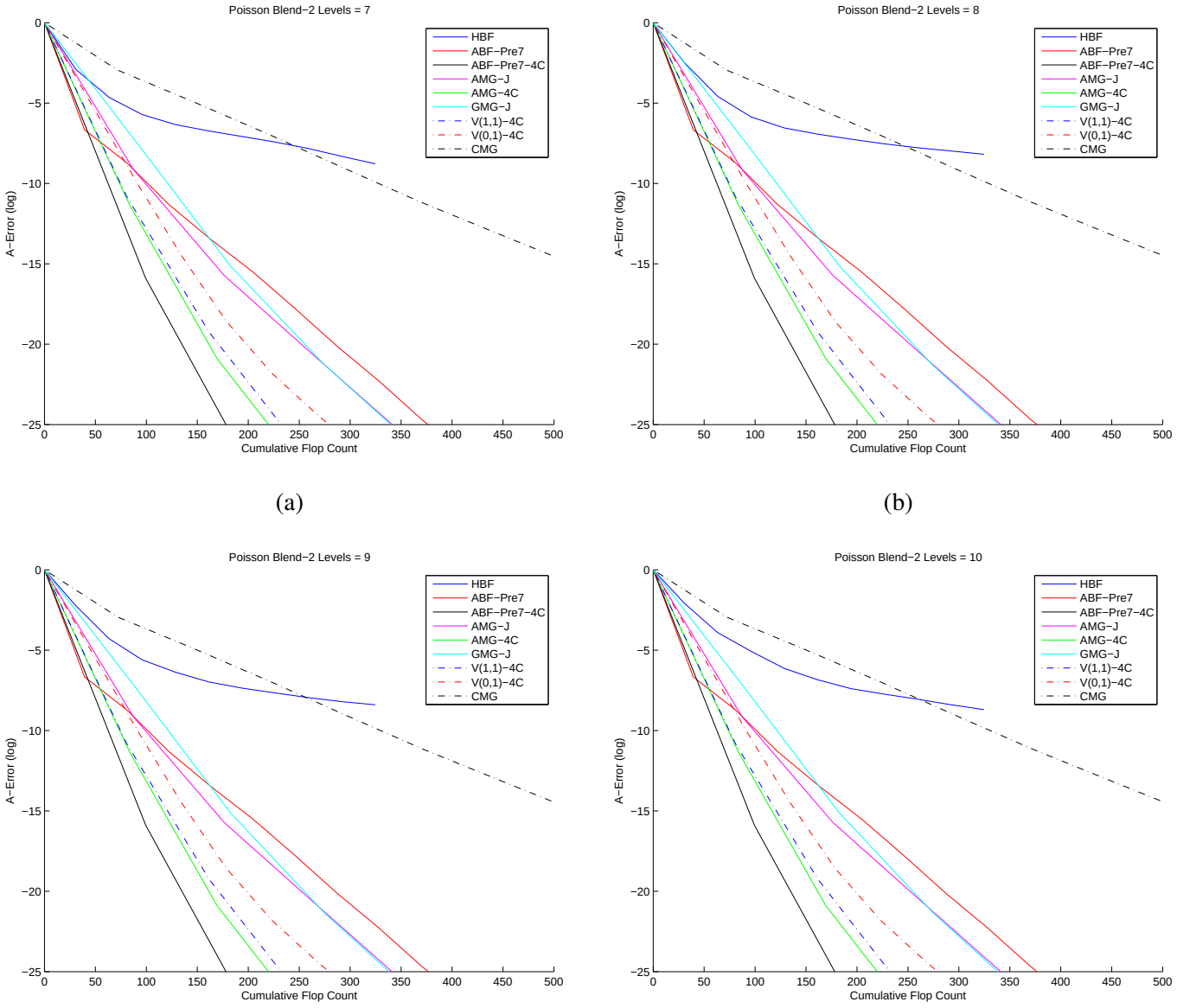


Figure 130: Log A-error vs. flops, Poisson blend, 2048×2048 image

	$L=1$				$L=2$				$L=3$				$L=4$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.70	0.36	43.7	0.83	0.82	0.20	35.6	0.57	0.73	0.31	34.6	0.89	0.72	0.33	34.6	0.94
ABF-Pre7	0.01	4.79	47.7	10.06	0.04	3.19	42.8	7.46	0.06	2.89	42.7	6.77	0.06	2.75	42.9	6.41
ABF-Pre7-4C	0.00	5.91	77.5	7.62	0.01	5.09	96.4	5.27	0.01	4.92	103.0	4.77	0.01	4.89	105.3	4.64
AMG-J	0.01	4.38	78.8	5.56	0.02	3.97	88.4	4.49	0.02	3.84	92.4	4.15	0.02	3.75	93.9	3.99
AMG-4C	0.00	5.82	74.7	7.80	0.01	4.83	83.3	5.81	0.01	4.72	87.0	5.43	0.01	4.72	88.3	5.35
GMG-J	0.14	2.00	83.7	2.39	0.19	1.64	92.6	1.77	0.32	1.14	96.4	1.19	0.35	1.04	97.8	1.06
V(1,1)-4C	0.02	3.78	70.2	5.39	0.08	2.50	78.8	3.17	0.08	2.49	82.5	3.02	0.09	2.46	83.8	2.93
V(0,1)-4C	0.03	3.52	47.2	7.45	0.12	2.15	45.4	4.74	0.08	2.49	46.2	5.39	0.08	2.57	46.7	5.49
CMG	0.21	1.55	57.5	2.70	0.23	1.48	67.2	2.21	0.24	1.43	72.8	1.96	0.24	1.43	78.3	1.82
ABF-Pre7-W	0.00	9.77	75.3	12.97	0.00	9.57	115.1	8.31	0.00	9.52	167.3	5.69	0.00	9.52	239.5	3.98
ABF-Pre7-4C-W	0.00	10.97	135.4	8.10	0.00	10.85	302.7	3.58	0.00	10.84	498.6	2.18	0.00	10.84	750.7	1.44
AMG-4C-W	0.00	5.82	74.7	7.80	0.00	5.72	109.0	5.24	0.00	5.72	130.8	4.37	0.00	5.72	145.2	3.94

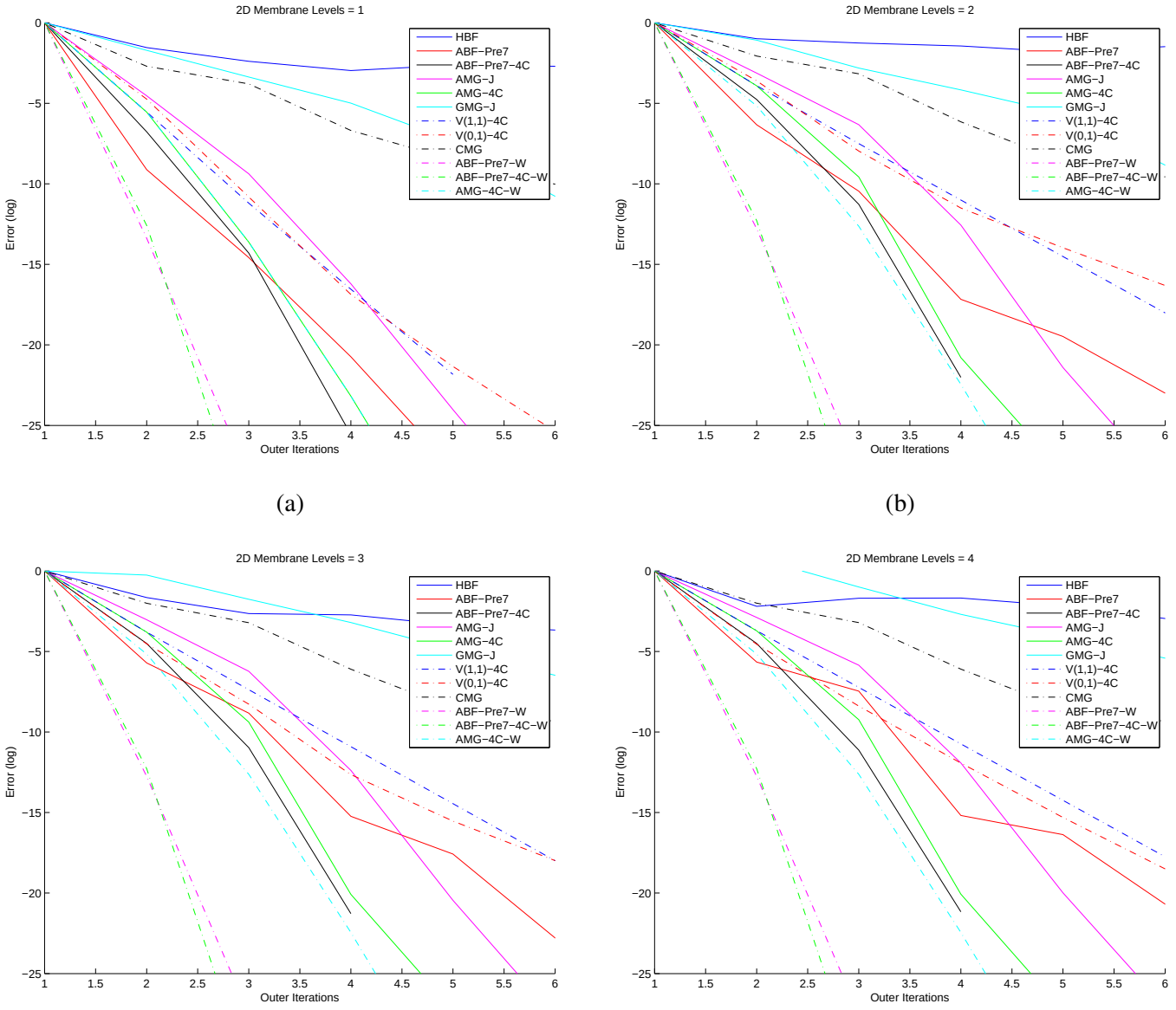
(a) empirical convergence results

	$L=1$						$L=2$						$L=3$						$L=4$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	23.11	11865.80	0.66	0.42	43.7	0.97	46.75	6239.95	0.74	0.29	35.6	0.83	88.33	1245.86	0.81	0.21	34.6	0.62	92.34	1036.12	0.81	0.21	34.6	0.60
ABF-Pre7	1.36	5105.17	0.08	2.56	47.7	5.37	1.37	1431.30	0.08	2.54	42.8	5.93	1.43	20.86	0.09	2.43	42.7	5.68	1.65	8.37	0.13	2.08	42.9	4.85
ABF-Pre7-4C	1.05	5105.17	0.01	4.51	77.5	5.81	1.10	1431.30	0.02	3.75	96.4	3.89	1.12	20.86	0.03	3.58	103.0	3.47	1.12	8.37	0.03	3.56	105.3	3.38
AMG-J	1.14	5078.45	0.03	3.41	78.8	4.33	1.22	1401.57	0.05	3.01	88.4	3.41	1.29	19.07	0.06	2.77	92.4	2.99	1.33	6.09	0.07	2.64	93.9	2.81
AMG-4C	1.07	5078.45	0.02	4.05	74.7	5.42	1.14	1401.57	0.03	3.40	83.3	4.08	1.18	19.07	0.04	3.20	87.0	3.68	1.19	6.09	0.04	3.15	88.3	3.56
GMG-J	2.58	11865.80	0.23	1.46	83.7	1.74	3.02	6239.95	0.27	1.31	92.6	1.41	3.53	1245.86	0.31	1.19	96.4	1.23	4.19	1036.12	0.34	1.07	97.8	1.09
V(1,1)-4C	1.07	5078.45	0.02	4.05	70.2	5.77	1.14	1401.57	0.03	3.40	78.8	4.32	1.18	19.07	0.04	3.20	82.5	3.89	1.19	6.09	0.04	3.15	83.8	3.75
V(0,1)-4C	1.21	5078.45	0.05	3.05	47.2	6.47	1.42	1401.57	0.09	2.43	45.4	5.35	1.50	19.07	0.10	2.29	46.2	4.95	1.49	6.09	0.10	2.31	46.7	4.95
CMG	5.99	483.37	0.42	0.87	57.5	1.51	5.90	137.64	0.42	0.88	67.2	1.30	6.20	58.05	0.43	0.85	72.8	1.17	6.20	17.57	0.43	0.85	78.3	1.09
ABF-Pre7-W	1.00	5105.17	0.00	8.91	75.3	11.83	1.00	1431.30	0.00	8.94	115.1	7.77	1.00	20.86	0.00	8.95	167.3	5.35	1.00	8.37	0.00	8.95	239.5	3.74
ABF-Pre7-4C-W	1.00	5105.17	0.00	7.73	135.4	5.71	1.00	1431.30	0.00	7.58	302.7	2.50	1.00	20.86	0.00	7.58	498.6	1.52	1.00	8.37	0.00	7.58	750.7	1.01
AMG-4C-W	1.07	5078.45	0.02	4.05	74.7	5.42	1.08	1401.57	0.02	3.94	109.0	3.61	1.08	19.07	0.02	3.94	130.8	3.01	1.08	6.09	0.02	3.94	145.2	2.71

Original condition number =5077.9

(b) theoretical convergence results

Table 111: 2D membrane, 33×33 grid



(a) (b)

(c) (d)

Figure 131: Log error vs. its, 2D membrane, 33×33 grid

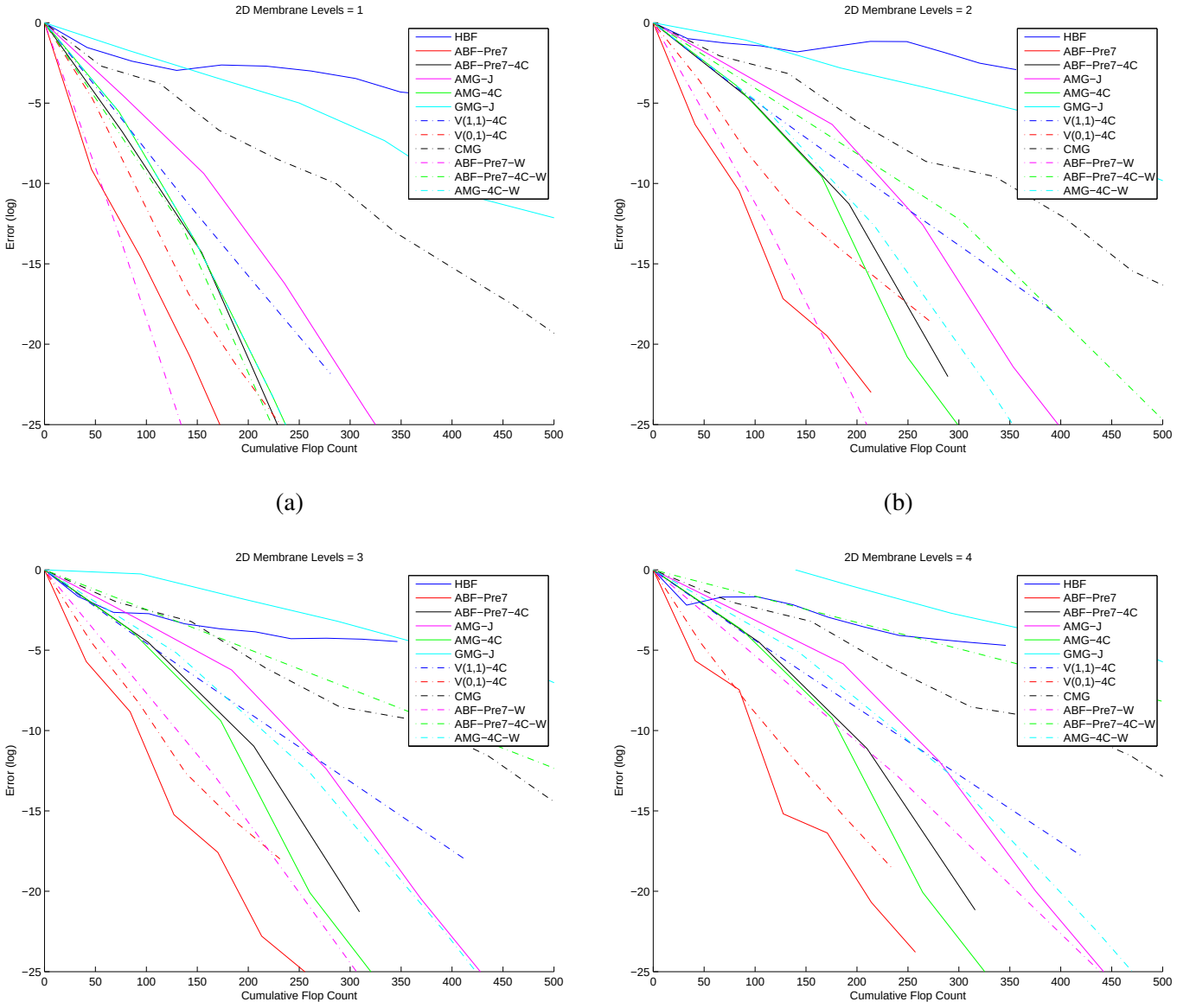
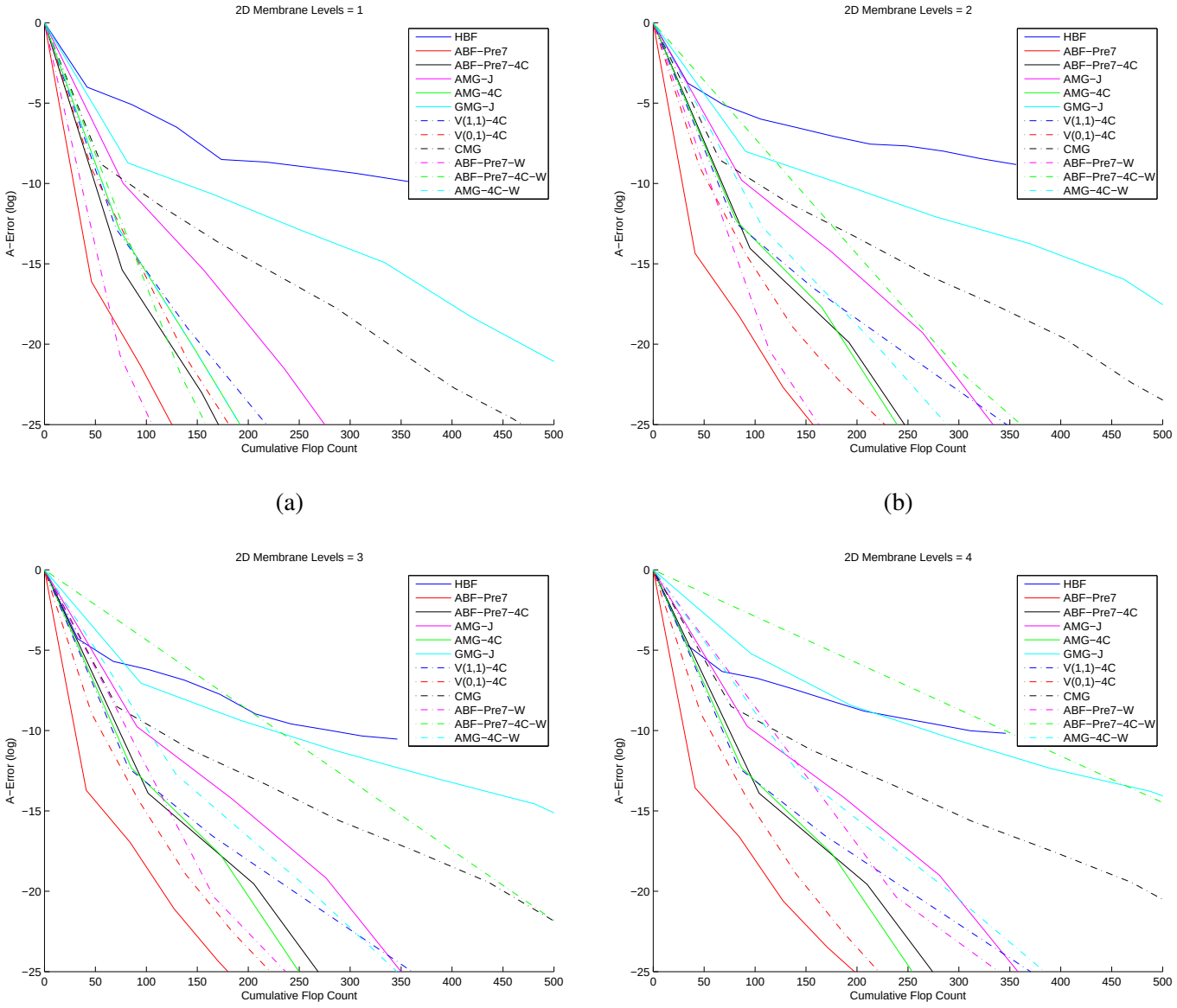


Figure 132: Log error vs. flops, 2D membrane, 33×33 grid



(a) (b)

(c) (d)

Figure 133: Log A-error vs. flops, 2D membrane, 33×33 grid

	$L=4$				$L=5$				$L=6$				$L=7$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.91	0.10	33.3	0.29	0.84	0.17	33.1	0.52	0.80	0.22	33.1	0.66	0.79	0.23	33.1	0.70
ABF-Pre7	0.03	3.38	41.7	8.12	0.06	2.73	41.6	6.57	0.10	2.27	41.7	5.44	0.12	2.16	41.7	5.19
ABF-Pre7-4C	0.01	4.29	100.6	4.26	0.01	4.28	100.9	4.25	0.01	4.31	101.0	4.27	0.01	4.33	101.0	4.29
AMG-J	0.02	3.69	90.4	4.09	0.02	3.69	90.5	4.08	0.03	3.40	90.6	3.75	0.05	3.01	90.6	3.32
AMG-4C	0.01	4.30	84.8	5.07	0.02	4.15	84.9	4.89	0.02	3.98	85.0	4.68	0.03	3.65	85.0	4.29
GMG-J	0.63	0.45	94.3	0.48	0.68	0.38	94.4	0.40	0.70	0.36	94.4	0.38	0.73	0.32	94.5	0.33
V(1,1)-4C	0.03	3.58	80.5	4.45	0.03	3.41	80.6	4.24	0.04	3.20	80.6	3.97	0.05	2.97	80.7	3.68
V(0,1)-4C	0.05	2.92	44.7	6.53	0.06	2.85	44.7	6.38	0.07	2.65	44.7	5.94	0.10	2.29	44.7	5.12
CMG	0.25	1.37	69.3	1.98	0.25	1.40	69.2	2.03	0.25	1.38	69.8	1.98	0.25	1.38	70.3	1.97
ABF-Pre7-W	0.00	10.05	204.9	4.90	0.00	10.05	239.6	4.20	0.00	10.05	279.1	3.60	0.00	10.05	319.5	3.15
ABF-Pre7-4C-W	0.00	8.50	623.9	1.36	0.00	8.50	778.6	1.09	0.00	8.50	938.1	0.91	0.00	8.50	1098.5	0.77
AMG-4C-W	0.01	4.61	132.4	3.48	0.01	4.61	136.3	3.38	0.01	4.61	138.5	3.33	0.01	4.61	139.6	3.30

(a) empirical convergence results

	$L=4$						$L=5$						$L=6$						$L=7$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	80.16	399001.14	0.80	0.22	33.3	0.67	103.03	176087.25	0.82	0.20	33.1	0.60	96.69	23215.87	0.82	0.20	33.1	0.62	93.75	314.25	0.81	0.21	33.1	0.63
ABF-Pre7	1.45	4544.28	0.09	2.39	41.7	5.73	1.56	1093.22	0.11	2.20	41.6	5.28	1.72	12.89	0.13	2.00	41.7	4.81	1.93	5.15	0.16	1.81	41.7	4.35
ABF-Pre7-4C	1.15	4544.28	0.03	3.36	100.6	3.34	1.16	1093.22	0.04	3.28	100.9	3.26	1.15	12.89	0.03	3.35	101.0	3.32	1.15	5.15	0.03	3.36	101.0	3.33
AMG-J	1.14	4404.30	0.03	3.42	90.4	3.78	1.26	1034.92	0.06	2.84	90.5	3.14	1.44	11.80	0.09	2.39	90.6	2.64	1.69	6.09	0.13	2.04	90.6	2.25
AMG-4C	1.09	4404.30	0.02	3.79	84.8	4.47	1.14	1034.92	0.03	3.44	84.9	4.05	1.21	11.80	0.05	3.05	85.0	3.59	1.29	6.09	0.06	2.75	85.0	3.24
GMG-J	4.62	399001.14	0.37	1.01	94.3	1.07	5.33	176087.25	0.40	0.93	94.4	0.98	5.62	23215.87	0.41	0.90	94.4	0.95	5.30	314.25	0.39	0.93	94.5	0.99
V(1,1)-4C	1.09	4404.30	0.02	3.79	80.5	4.71	1.14	1034.92	0.03	3.44	80.6	4.27	1.21	11.80	0.05	3.05	80.6	3.78	1.29	6.09	0.06	2.75	80.7	3.41
V(0,1)-4C	1.57	4404.30	0.11	2.18	44.7	4.88	1.83	1034.92	0.15	1.90	44.7	4.26	2.02	11.80	0.17	1.75	44.7	3.92	2.22	6.09	0.20	1.63	44.7	3.65
CMG	6.76	1307.53	0.44	0.81	69.3	1.17	7.27	394.21	0.46	0.78	69.2	1.13	7.30	188.08	0.46	0.78	69.8	1.11	7.39	43.05	0.46	0.77	70.3	1.10
ABF-Pre7-W	-	-	-	-	204.9	-	-	-	-	-	239.6	-	-	-	-	-	279.1	-	-	-	-	-	319.5	-
ABF-Pre7-4C-W	-	-	-	-	623.9	-	-	-	-	-	778.6	-	-	-	-	-	938.1	-	-	-	-	-	1098.5	-
AMG-4C-W	-	-	-	-	132.4	-	-	-	-	-	136.3	-	-	-	-	-	138.5	-	-	-	-	-	139.6	-

Original condition number =105809.6

(b) theoretical convergence results

Table 112: 2D membrane, 256×256 grid

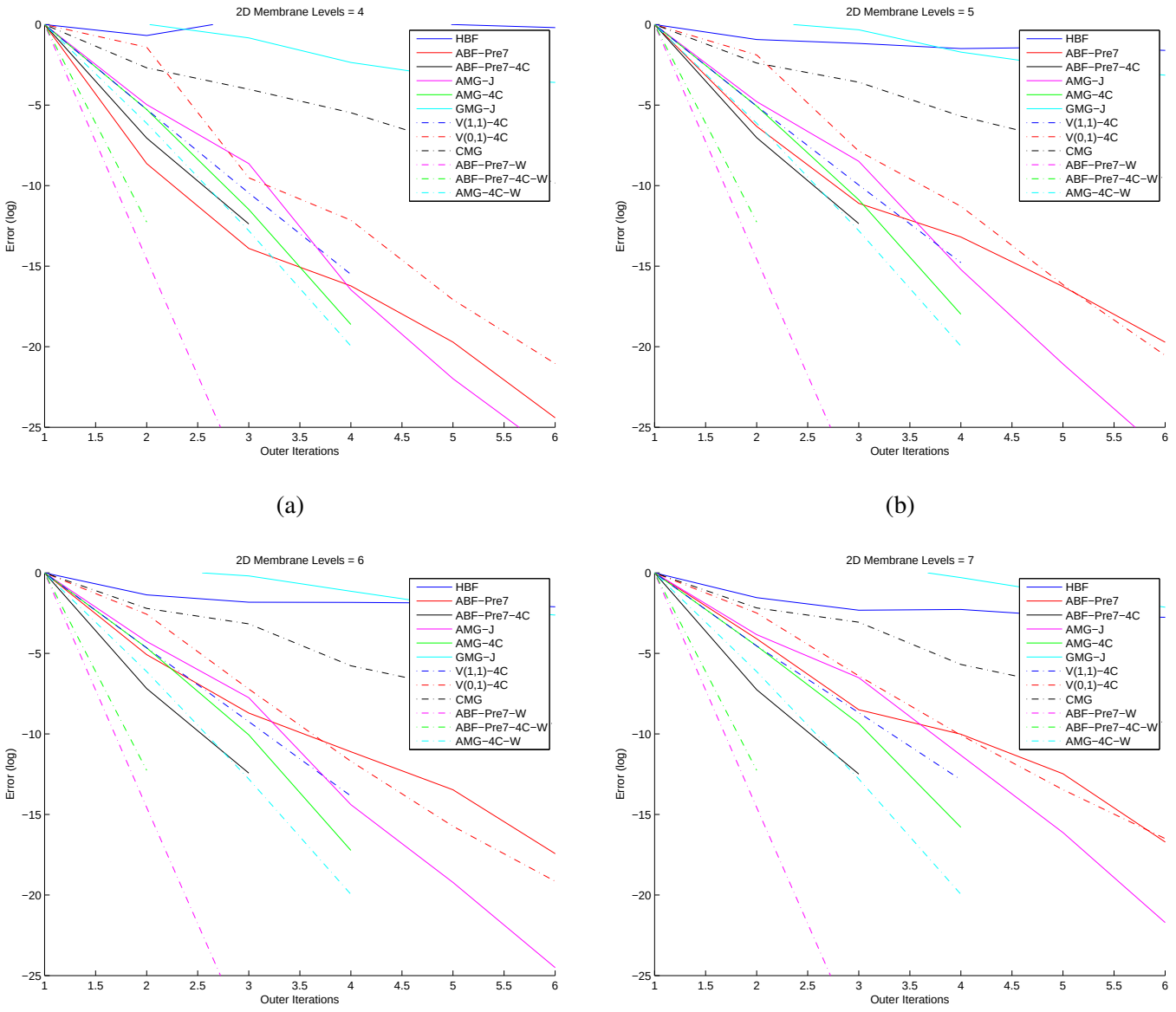
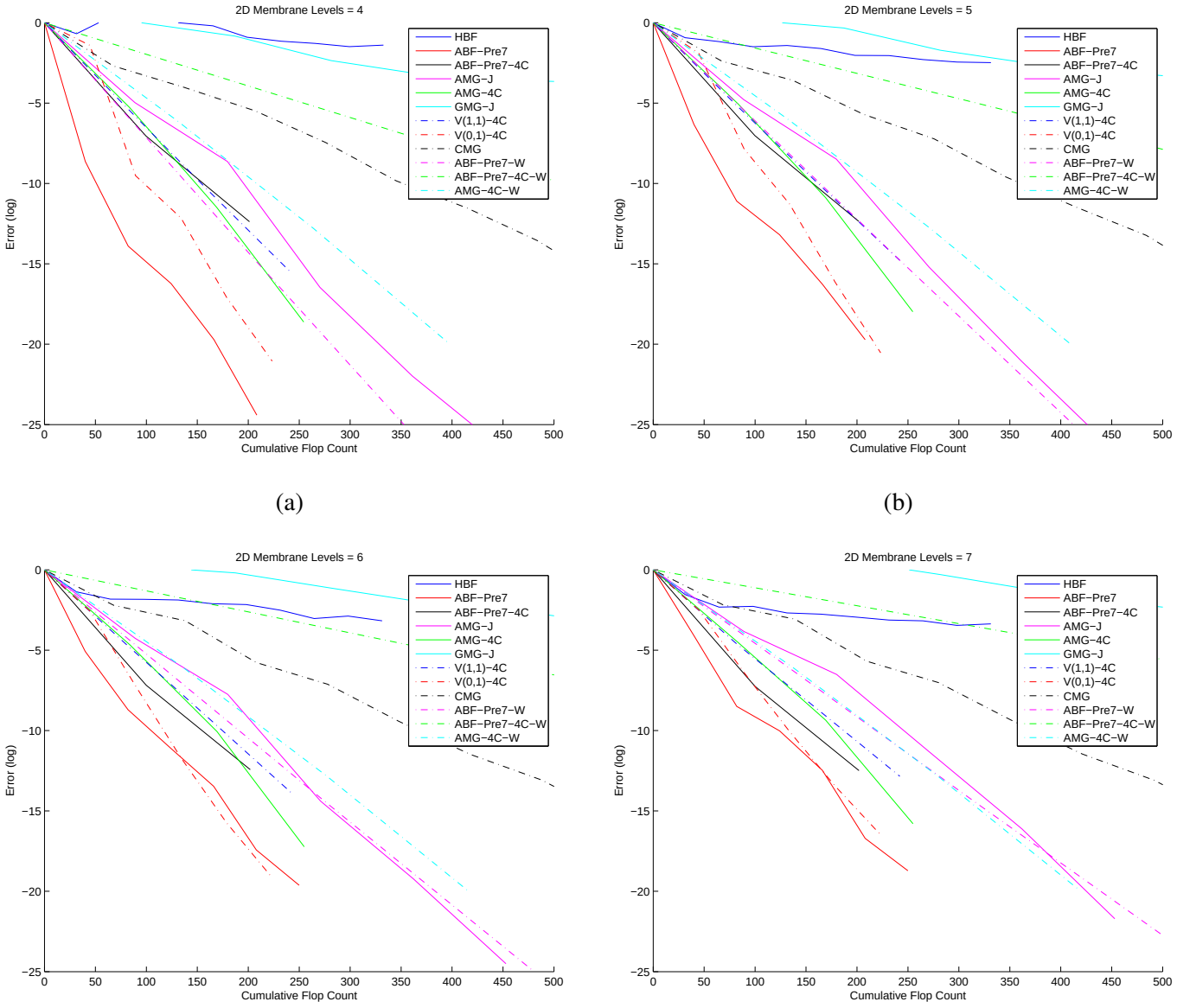


Figure 134: Log error vs. its, 2D membrane, 256×256 grid



(a) (b)

(c) (d)

Figure 135: Log error vs. flops, 2D membrane, 256×256 grid

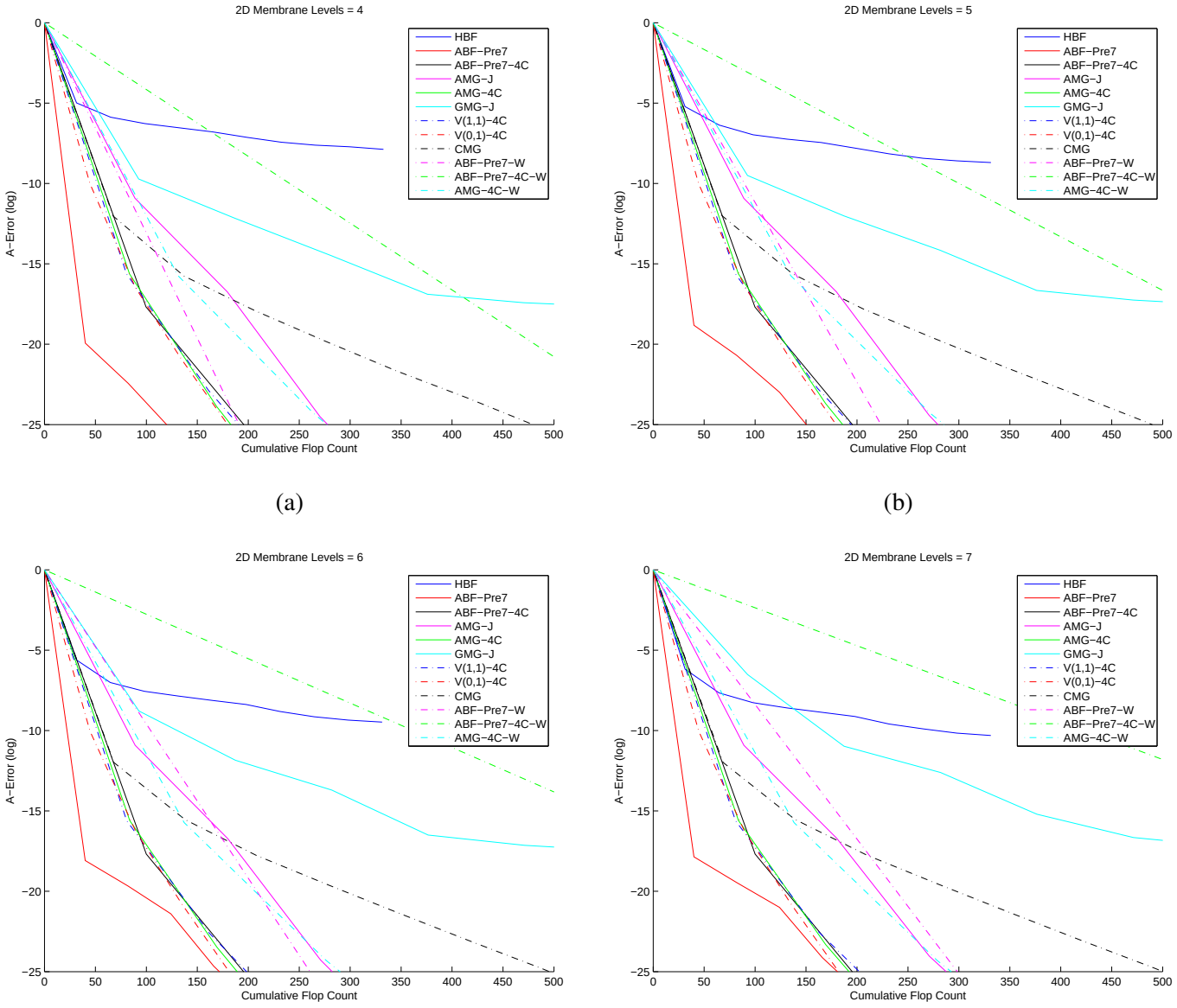
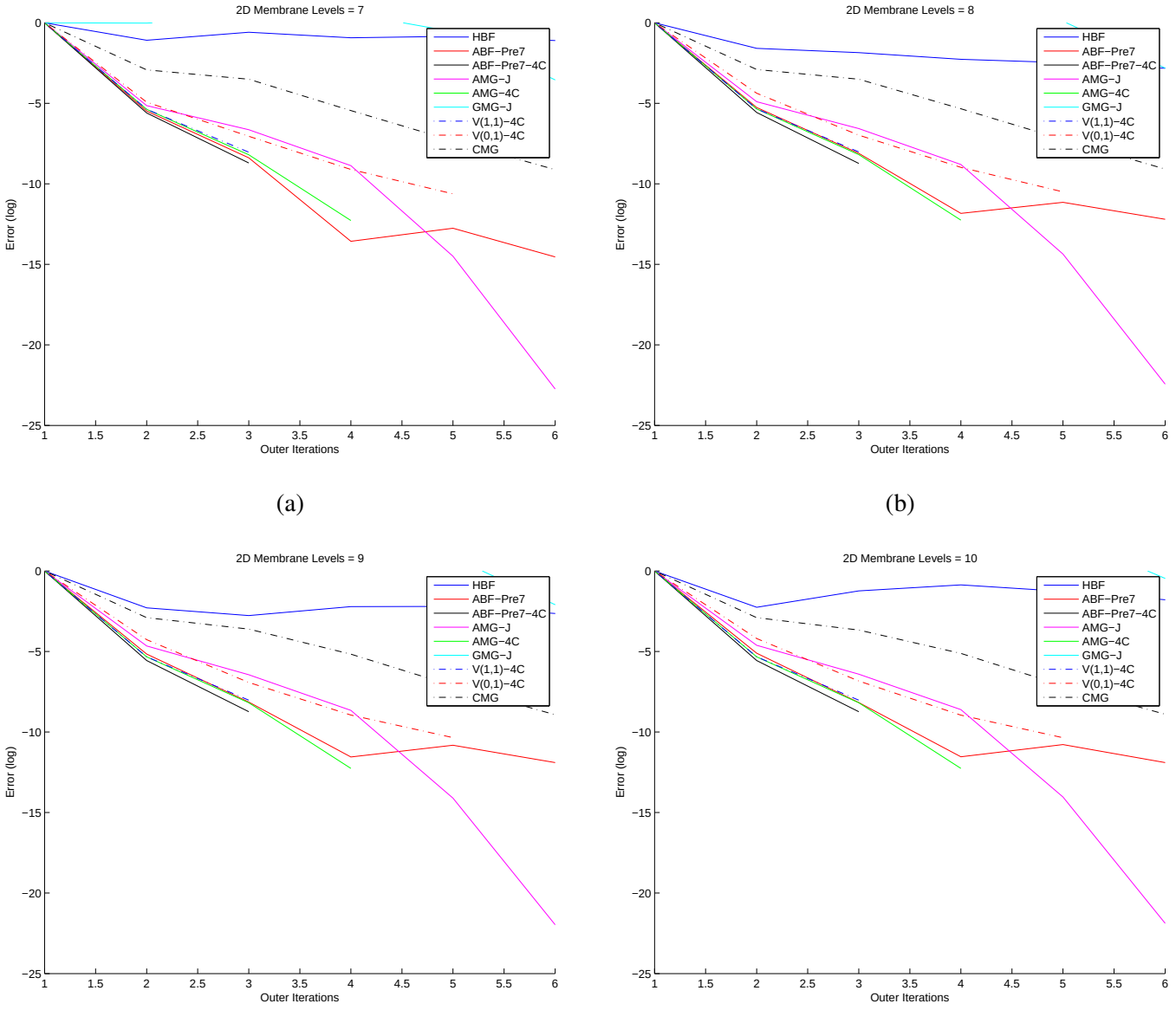


Figure 136: Log A-error vs. flops, 2D membrane, 256×256 grid

	$L = 7$				$L = 8$				$L = 9$				$L = 10$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.85	0.16	33.1	0.49	0.75	0.28	33.1	0.85	0.70	0.36	33.1	1.09	0.73	0.32	33.1	0.96
ABF-Pre7	0.13	2.02	41.6	4.85	0.18	1.69	41.6	4.07	0.19	1.65	41.6	3.96	0.19	1.65	41.6	3.97
ABF-Pre7-4C	0.05	3.02	101.0	2.99	0.05	3.03	101.0	3.00	0.05	3.03	101.0	3.00	0.05	3.03	101.0	3.00
AMG-J	0.04	3.15	90.6	3.48	0.04	3.11	90.6	3.43	0.05	3.05	90.6	3.36	0.05	3.03	90.6	3.35
AMG-4C	0.06	2.84	85.0	3.34	0.06	2.83	85.0	3.33	0.06	2.83	85.0	3.33	0.06	2.83	85.0	3.33
GMG-J	0.51	0.67	94.4	0.71	0.51	0.67	94.4	0.71	0.51	0.67	94.4	0.71	0.58	0.54	94.5	0.57
V(1,1)-4C	0.06	2.78	80.7	3.45	0.06	2.78	80.7	3.45	0.06	2.78	80.7	3.45	0.06	2.78	80.7	3.45
V(0,1)-4C	0.16	1.84	44.7	4.12	0.16	1.82	44.7	4.07	0.17	1.79	44.7	4.02	0.17	1.79	44.7	4.02
CMG	0.30	1.21	69.4	1.74	0.30	1.21	69.6	1.74	0.30	1.22	69.6	1.75	0.30	1.22	69.8	1.75

(a) empirical convergence results

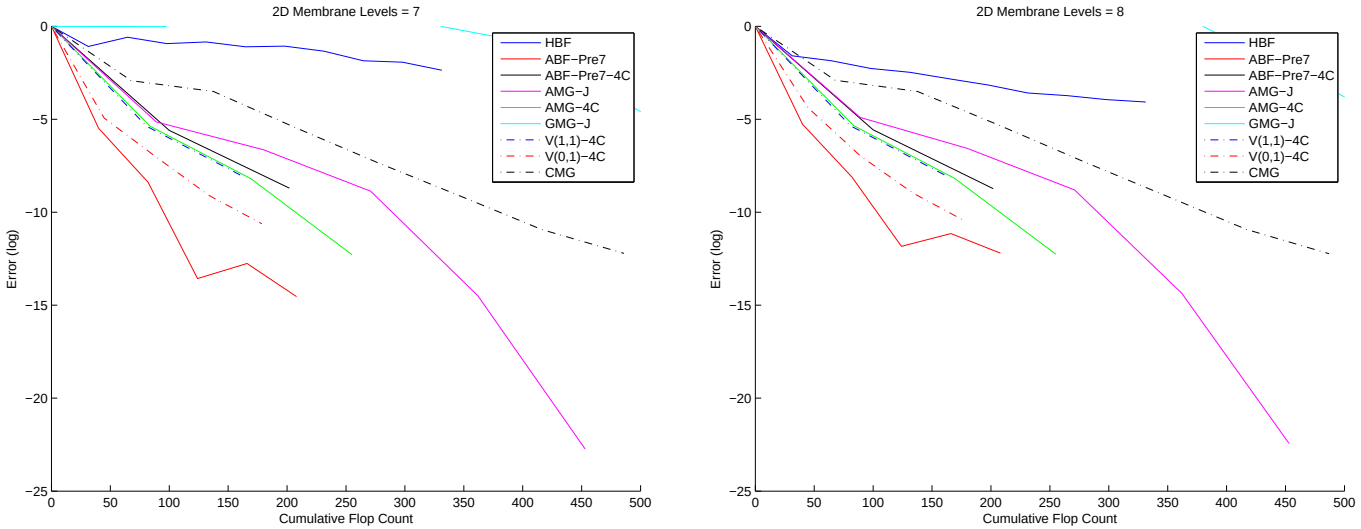
Table 113: 2D membrane, 2048×2048 grid



(a) (b)

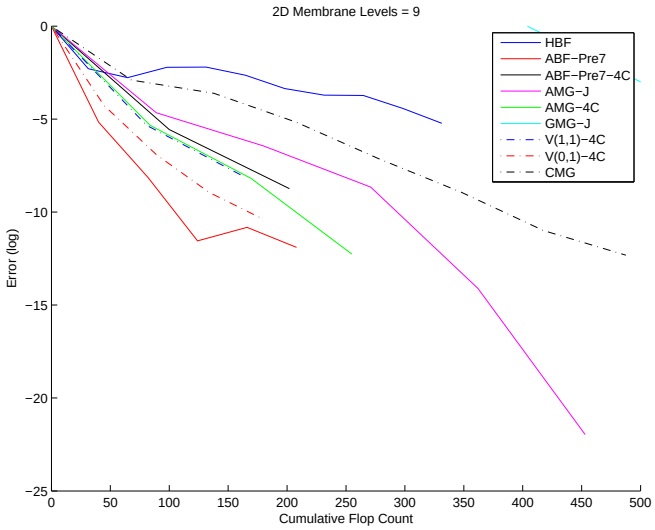
(c) (d)

Figure 137: Log error vs. its, 2D membrane, 2048×2048 grid

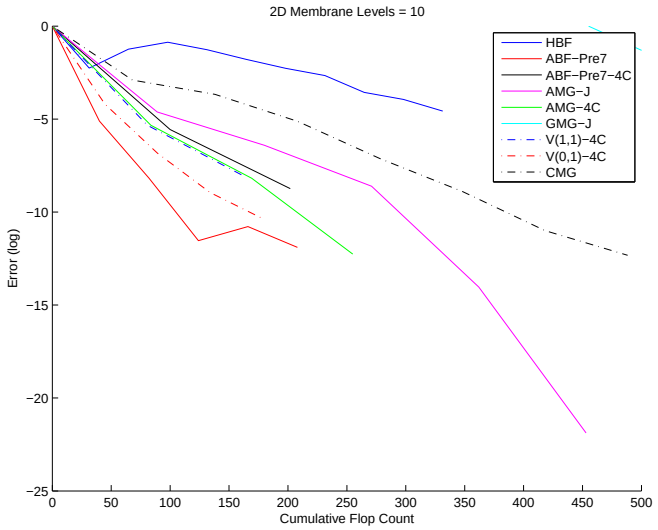


(a)

(b)

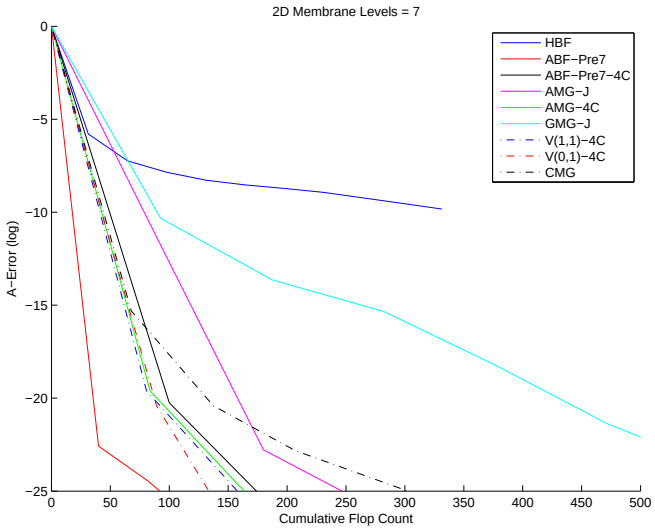


(c)

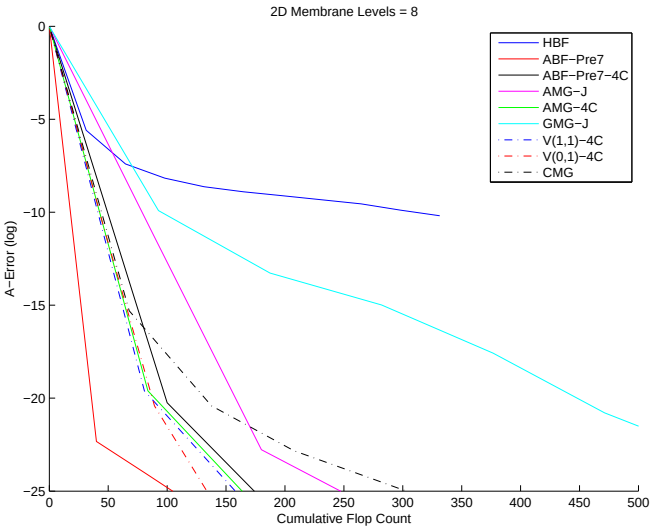


(d)

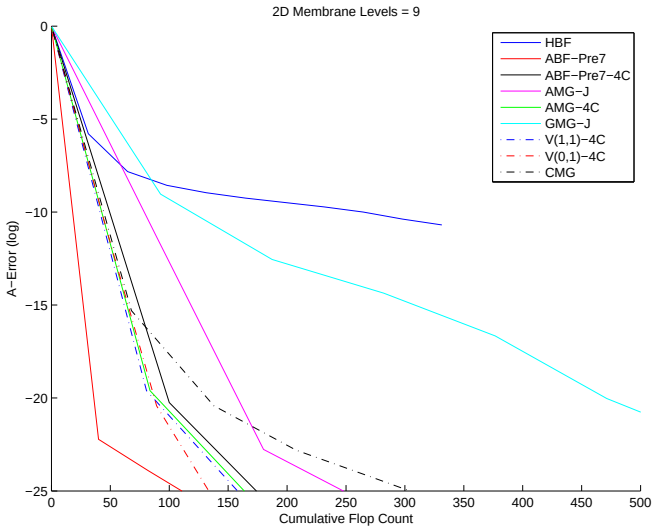
Figure 138: Log error vs. flops, 2D membrane, 2048×2048 grid



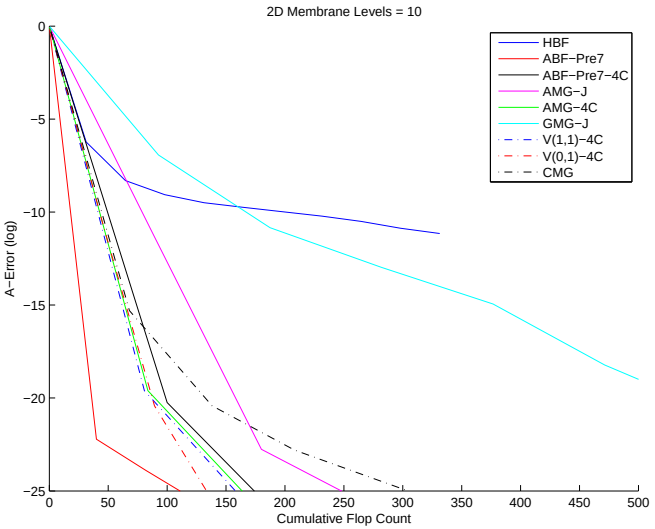
(a)



(b)



(c)



(d)

Figure 139: Log A-error vs. flops, 2D membrane, 2048×2048 grid

	$L=1$				$L=2$				$L=3$				$L=4$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.80	0.22	35.1	0.64	0.77	0.27	32.6	0.82	0.69	0.37	32.5	1.15	0.69	0.36	32.6	1.12
ABF-Pre7	0.02	3.79	41.3	9.17	0.02	3.75	40.4	9.28	0.02	3.75	40.8	9.20	0.02	3.75	40.9	9.16
ABF-Pre7-4C	0.00	8.23	70.8	11.63	0.00	7.26	92.4	7.85	0.00	7.23	98.4	7.35	0.00	7.23	100.0	7.24
AMG-J	0.05	2.96	71.5	4.14	0.05	2.93	84.2	3.48	0.05	2.93	87.9	3.33	0.05	2.93	88.9	3.29
AMG-4C	0.01	4.97	68.2	7.28	0.01	4.96	80.2	6.19	0.01	4.96	83.7	5.93	0.01	4.96	84.7	5.86
GMG-J	0.19	1.64	74.0	2.21	0.31	1.18	87.3	1.35	0.30	1.20	91.2	1.32	0.30	1.20	92.3	1.30
V(1,1)-4C	0.02	4.16	63.9	6.51	0.02	4.16	75.8	5.49	0.02	4.16	79.4	5.25	0.02	4.16	80.3	5.18
V(0,1)-4C	0.08	2.47	40.9	6.04	0.09	2.46	43.1	5.72	0.09	2.46	44.2	5.57	0.09	2.46	44.5	5.52
ABF-Pre7-W	0.00	7.55	62.8	12.02	0.00	7.55	101.1	7.47	0.00	7.55	141.3	5.34	0.00	7.55	183.0	4.13
ABF-Pre7-4C-W	0.00	16.45	121.8	13.51	0.00	16.49	280.1	5.89	0.00	16.49	440.3	3.74	0.00	16.49	602.0	2.74
AMG-4C-W	0.01	4.97	68.2	7.28	0.01	4.97	103.2	4.81	0.01	4.97	121.6	4.08	0.01	4.97	131.1	3.79

(a) empirical convergence results

	$L=1$						$L=2$						$L=3$						$L=4$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	5.60	9851.04	0.41	0.90	35.1	2.57	18.68	476.28	0.62	0.47	32.6	1.45	19.72	58.41	0.63	0.46	32.5	1.41	25.46	35.93	0.67	0.40	32.6	1.23
ABF-Pre7	1.49	10771.75	0.10	2.32	41.3	5.61	1.50	7224.35	0.10	2.30	40.4	5.68	1.50	2403.16	0.10	2.30	40.8	5.63	1.50	302.11	0.10	2.30	40.9	5.61
ABF-Pre7-4C	1.10	10771.75	0.02	3.72	70.8	5.26	1.15	7224.35	0.04	3.34	92.4	3.61	1.15	2403.16	0.04	3.33	98.4	3.38	1.15	302.11	0.04	3.33	100.0	3.33
AMG-J	2.09	10001.93	0.18	1.70	71.5	2.38	2.04	7352.48	0.18	1.74	84.2	2.06	2.03	2382.09	0.18	1.74	87.9	1.98	2.03	280.01	0.18	1.74	88.9	1.96
AMG-4C	1.24	10001.93	0.05	2.94	68.2	4.31	1.23	7352.48	0.05	2.95	80.2	3.68	1.23	2382.09	0.05	2.95	83.7	3.53	1.23	280.01	0.05	2.95	84.7	3.49
GMG-J	4.02	9851.04	0.33	1.10	74.0	1.48	7.74	476.28	0.47	0.75	87.3	0.86	8.40	58.41	0.49	0.72	91.2	0.79	8.29	35.93	0.48	0.72	92.3	0.79
V(1,1)-4C	1.24	10001.93	0.05	2.94	63.9	4.60	1.23	7352.48	0.05	2.95	75.8	3.89	1.23	2382.09	0.05	2.95	79.4	3.72	1.23	280.01	0.05	2.95	80.3	3.67
V(0,1)-4C	1.49	10001.93	0.10	2.31	40.9	5.66	1.52	7352.48	0.10	2.26	43.1	5.25	1.51	2382.09	0.10	2.28	44.2	5.15	1.51	280.01	0.10	2.28	44.5	5.11
ABF-Pre7-W	1.00	10771.75	0.00	8.06	62.8	12.83	1.00	7224.35	0.00	8.08	101.1	7.99	1.00	2403.16	0.00	8.08	141.3	5.72	1.00	302.11	0.00	8.08	183.0	4.42
ABF-Pre7-4C-W	1.00	10771.75	0.00	9.52	121.8	7.82	1.00	7224.35	0.00	9.53	280.1	3.40	1.00	2403.16	0.00	9.53	440.3	2.17	1.00	302.11	0.00	9.53	602.0	1.58
AMG-4C-W	1.24	10001.93	0.05	2.94	68.2	4.31	1.24	7352.48	0.05	2.94	103.2	2.85	1.24	2382.09	0.05	2.94	121.6	2.42	1.24	280.01	0.05	2.94	131.1	2.24

Original condition number =918.4

(b) theoretical convergence results

Table 114: Colorization, 32×32 image

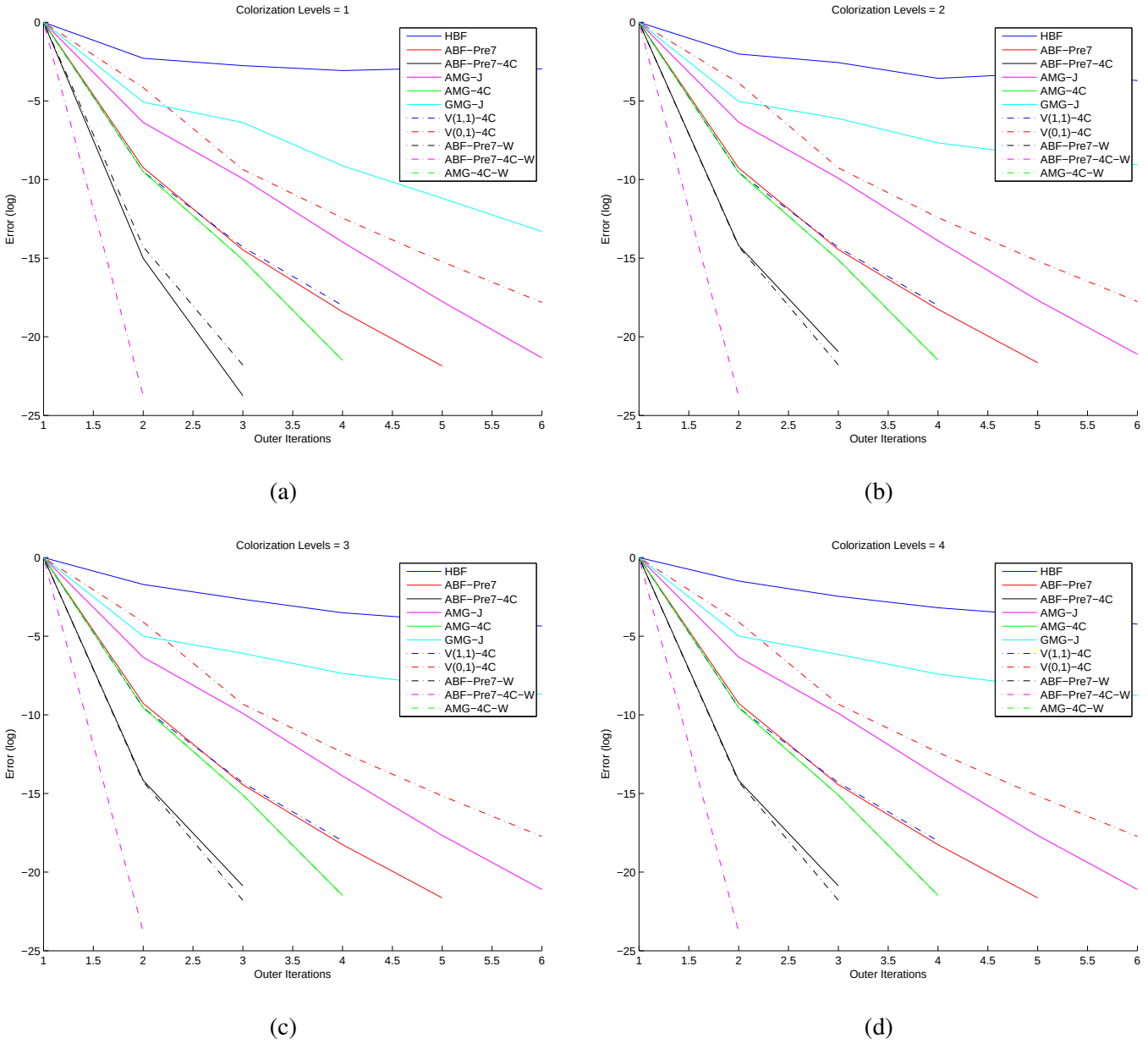


Figure 140: Log error vs. its, Colorization, 32×32 image

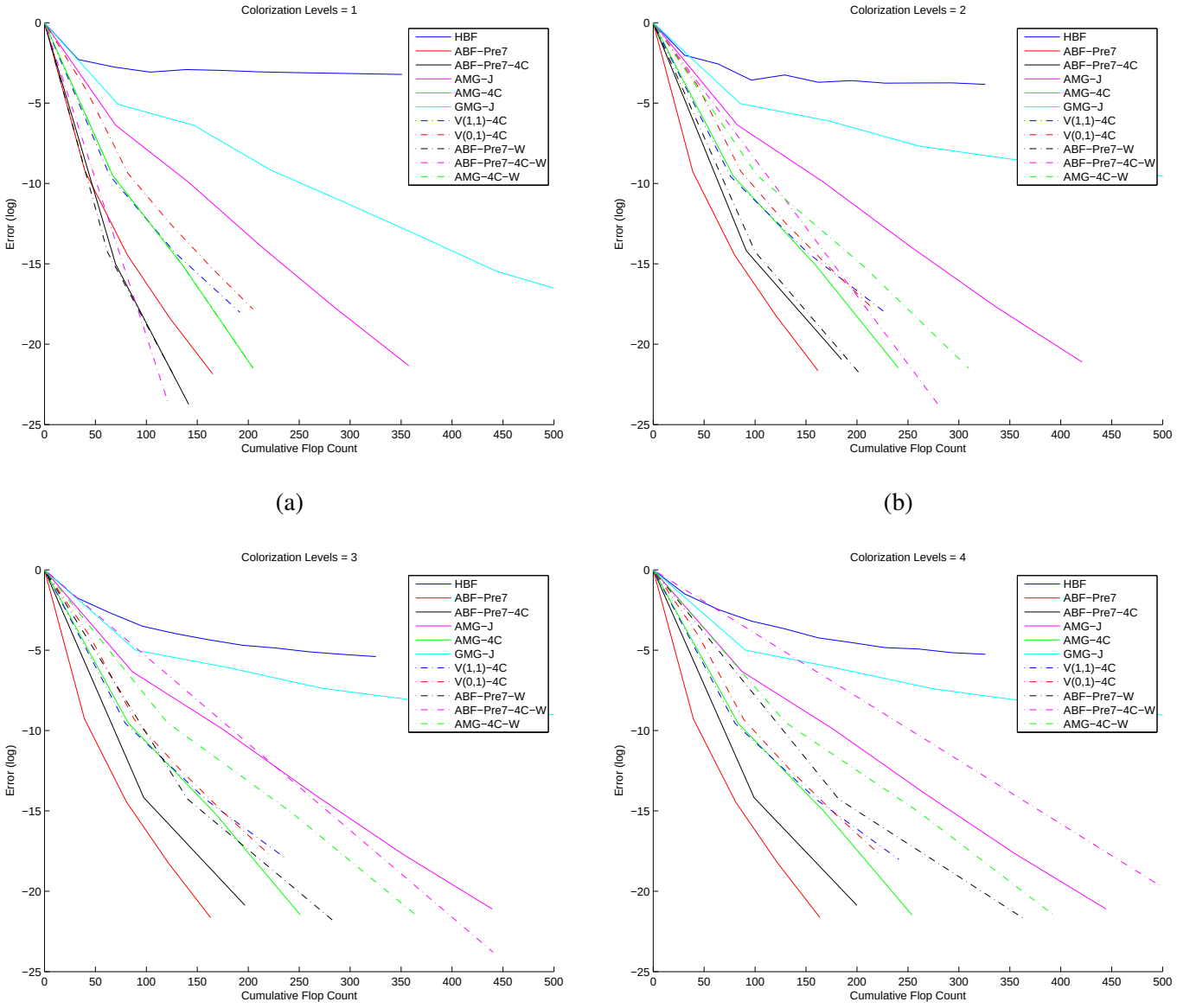


Figure 141: Log error vs. flops, Colorization, 32×32 image

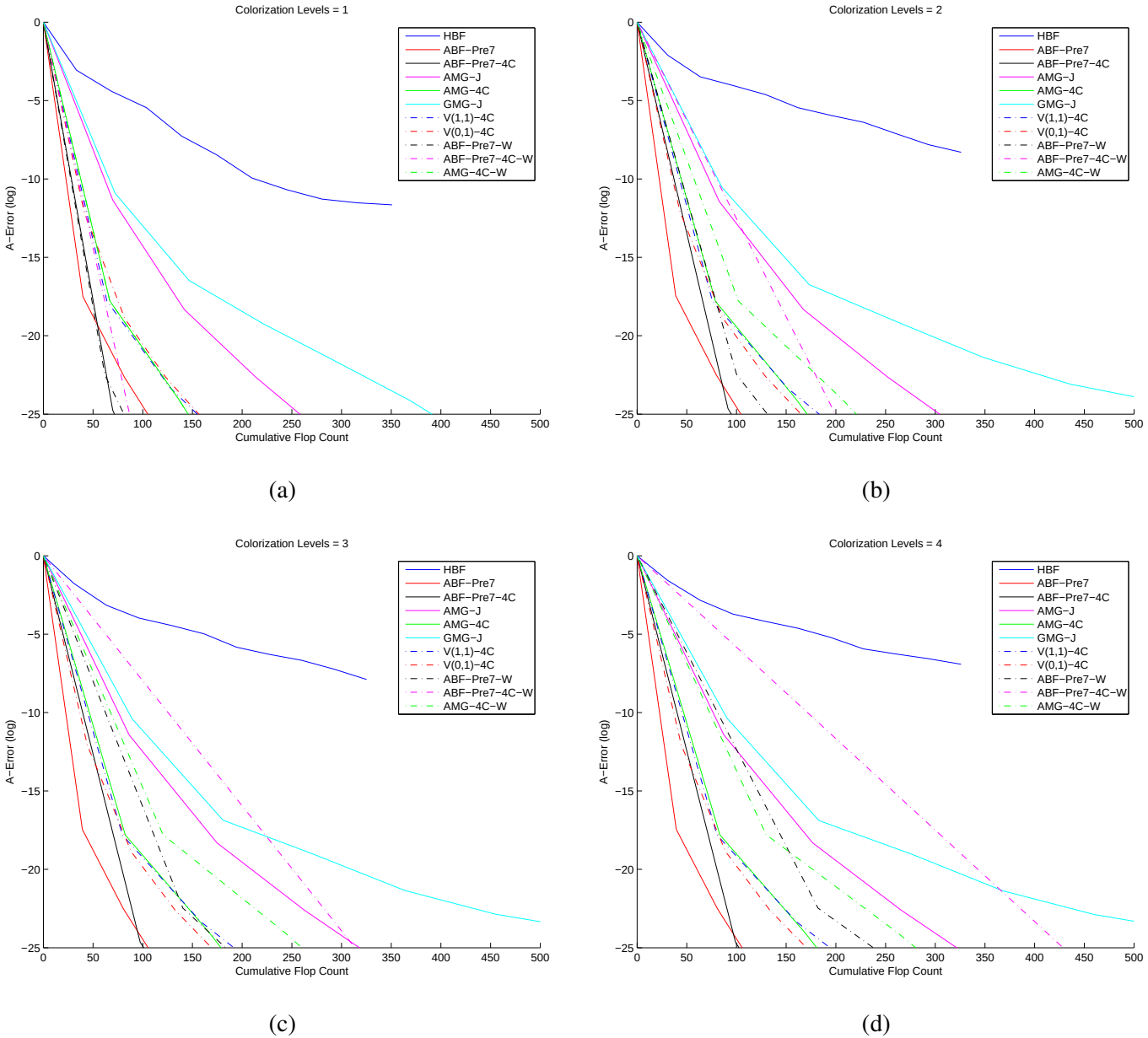


Figure 142: Log A-error vs. flops, Colorization, 32×32 image

	$L=1$				$L=2$				$L=3$				$L=4$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.82	0.20	35.1	0.57	0.80	0.22	32.6	0.68	0.76	0.27	32.5	0.84	0.74	0.30	32.6	0.92
ABF-Pre7	0.03	3.67	41.1	8.91	0.02	3.69	40.3	9.17	0.03	3.57	40.6	8.79	0.03	3.57	40.8	8.75
ABF-Pre7-4C	0.00	11.25	69.8	16.12	0.00	11.09	91.4	12.13	0.00	11.17	97.4	11.47	0.00	11.17	99.0	11.29
AMG-J	0.09	2.46	71.5	3.44	0.09	2.46	84.2	2.92	0.09	2.46	87.9	2.80	0.09	2.46	88.9	2.77
AMG-4C	0.01	4.93	67.9	7.26	0.01	4.75	79.8	5.95	0.01	4.70	83.4	5.64	0.01	4.70	84.3	5.58
GMG-J	0.37	1.00	74.0	1.34	0.34	1.07	87.3	1.22	0.32	1.15	91.2	1.26	0.28	1.28	92.2	1.39
V(1,1)-4C	0.01	4.86	63.9	7.60	0.01	4.67	75.8	6.16	0.01	4.63	79.4	5.83	0.01	4.63	80.3	5.76
V(0,1)-4C	0.11	2.23	40.9	5.46	0.12	2.12	43.1	4.92	0.12	2.09	44.2	4.73	0.12	2.09	44.5	4.68
ABF-Pre7-W	0.01	4.98	62.8	7.93	0.01	4.98	101.1	4.93	0.01	4.98	141.3	3.53	0.01	4.98	183.0	2.72
ABF-Pre7-4C-W	0.00	15.13	121.8	12.42	0.00	15.12	280.1	5.40	0.00	15.12	440.3	3.43	0.00	15.12	602.0	2.51
AMG-4C-W	0.01	4.93	67.9	7.26	0.01	4.89	102.9	4.76	0.01	4.89	121.3	4.03	0.01	4.89	130.8	3.74

(a) empirical convergence results

	$L=1$						$L=2$						$L=3$						$L=4$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	6.65	18605.14	0.44	0.82	35.1	2.33	24.25	20015.84	0.66	0.41	32.6	1.26	23.59	356.83	0.66	0.42	32.5	1.28	31.93	40.62	0.70	0.36	32.6	1.10
ABF-Pre7	2.71	24358.51	0.24	1.41	41.1	3.43	2.72	12245.40	0.25	1.41	40.3	3.49	3.05	8988.59	0.27	1.30	40.6	3.21	3.05	8718.89	0.27	1.30	40.8	3.20
ABF-Pre7-4C	1.17	24358.51	0.04	3.25	69.8	4.66	1.23	12245.40	0.05	2.97	91.4	3.25	1.22	8988.59	0.05	3.01	97.4	3.09	1.22	8718.89	0.05	3.01	99.0	3.04
AMG-J	3.15	13987.95	0.28	1.28	71.5	1.78	2.78	12421.41	0.25	1.39	84.2	1.65	3.12	7148.51	0.28	1.28	87.9	1.46	3.13	3337.84	0.28	1.28	88.9	1.44
AMG-4C	2.02	13987.95	0.17	1.75	67.9	2.58	2.32	12421.41	0.21	1.57	79.8	1.97	2.38	7148.51	0.21	1.54	83.4	1.85	2.38	3337.84	0.21	1.54	84.3	1.83
GMG-J	13.03	18605.14	0.57	0.57	74.0	0.77	10.17	20015.84	0.52	0.65	87.3	0.74	9.46	356.83	0.51	0.67	91.2	0.74	9.06	40.62	0.50	0.69	92.2	0.75
V(1,1)-4C	2.02	13987.95	0.17	1.75	63.9	2.74	2.32	12421.41	0.21	1.57	75.8	2.08	2.38	7148.51	0.21	1.54	79.4	1.95	2.38	3337.84	0.21	1.54	80.3	1.92
V(0,1)-4C	2.34	13987.95	0.21	1.56	40.9	3.82	2.66	12421.41	0.24	1.43	43.1	3.32	3.18	7148.51	0.28	1.27	44.2	2.87	3.18	3337.84	0.28	1.27	44.5	2.84
ABF-Pre7-W	1.09	24358.51	0.02	3.81	62.8	6.06	1.09	12245.40	0.02	3.81	101.1	3.76	1.09	8988.59	0.02	3.81	141.3	2.69	1.09	8718.89	0.02	3.81	183.0	2.08
ABF-Pre7-4C-W	1.01	24358.51	0.00	6.54	121.8	5.37	1.01	12245.40	0.00	6.53	280.1	2.33	1.01	8988.59	0.00	6.53	440.3	1.48	1.01	8718.89	0.00	6.53	602.0	1.09
AMG-4C-W	2.02	13987.95	0.17	1.75	67.9	2.58	1.95	12421.41	0.17	1.80	102.9	1.75	1.95	7148.51	0.17	1.80	121.3	1.48	1.95	3337.84	0.17	1.80	130.8	1.38

Original condition number =2885.9

(b) theoretical convergence results

Table 115: Colorization, 32×32 image

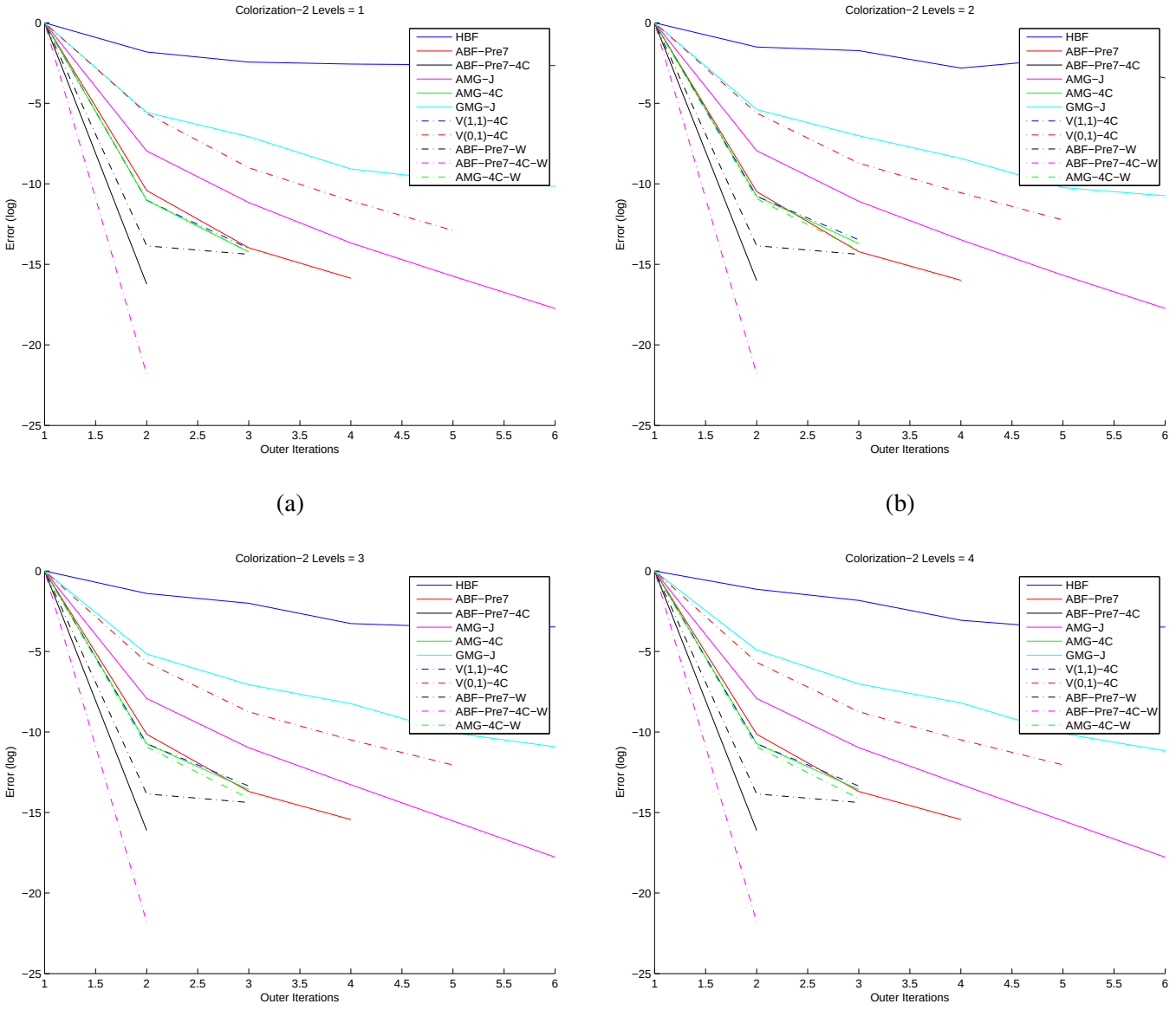


Figure 143: Log error vs. its, Colorization, 32×32 image

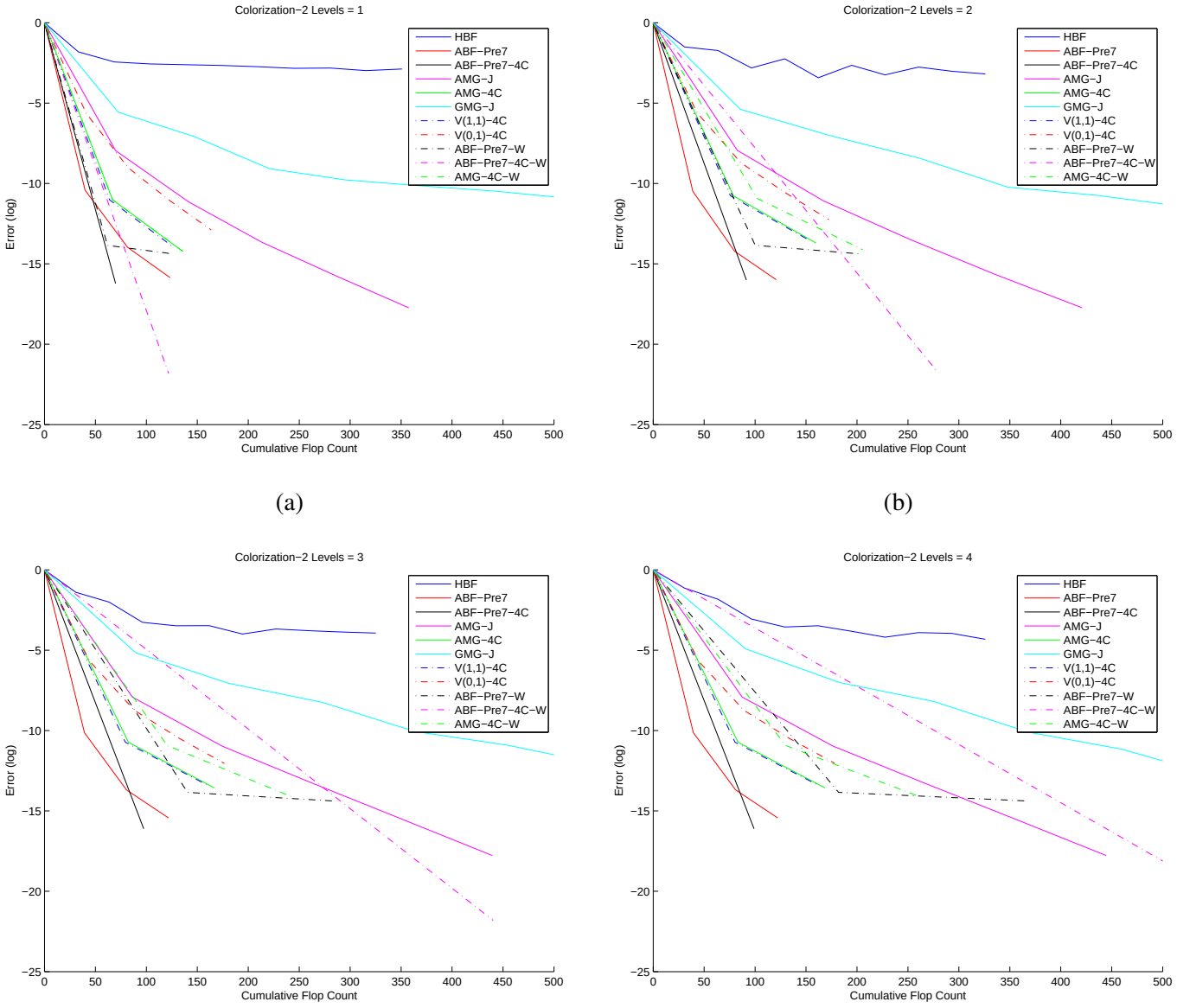


Figure 144: Log error vs. flops, Colorization, 32×32 image

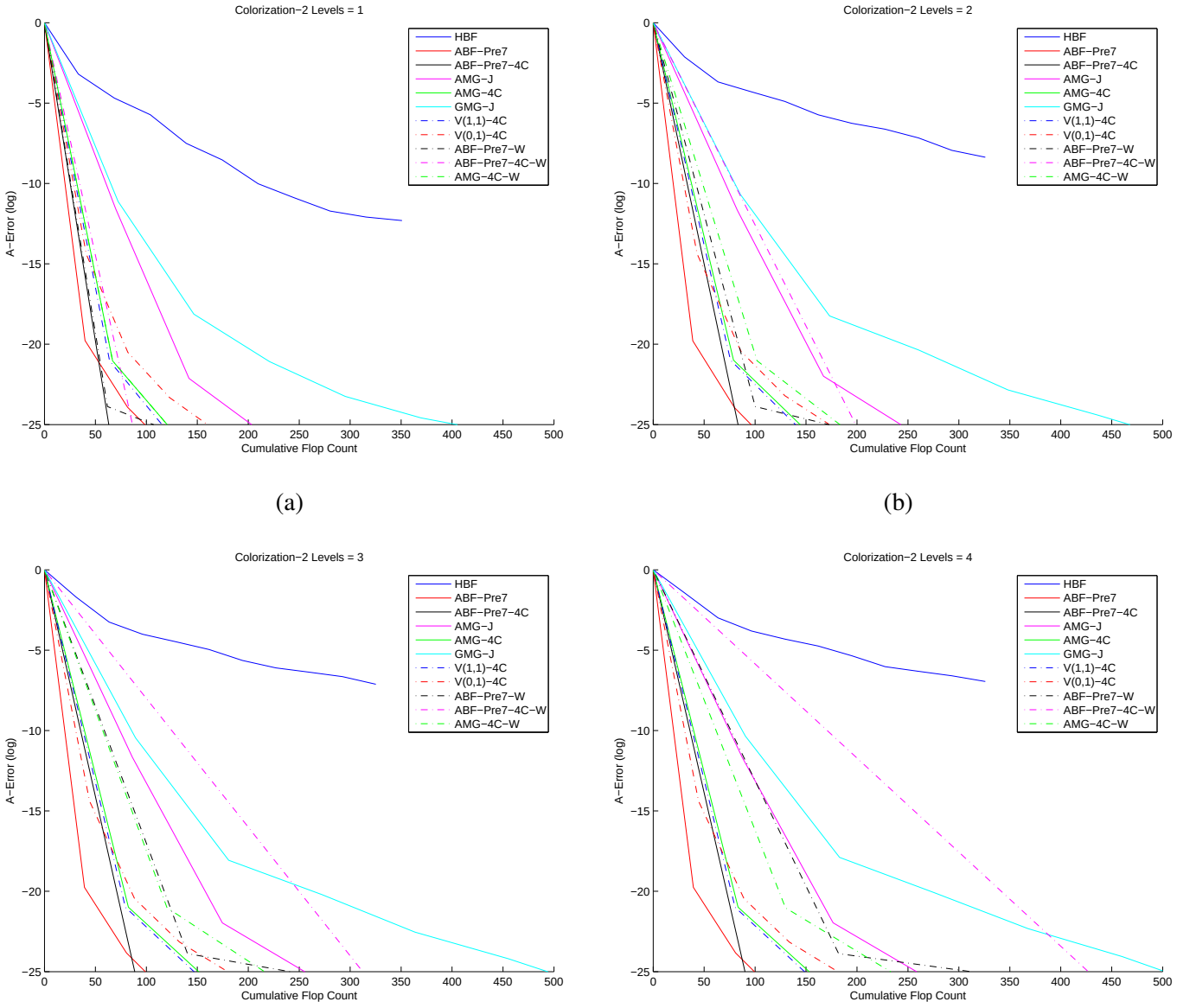


Figure 145: Log A-error vs. flops, Colorization, 32×32 image

Algorithm	$L=4$				$L=5$				$L=6$				$L=7$			
	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.89	0.12	32.8	0.37	0.75	0.29	32.7	0.89	0.70	0.36	32.7	1.11	0.68	0.39	32.7	1.18
ABF-Pre7	0.02	4.09	40.9	10.00	0.02	4.07	40.9	9.96	0.02	4.08	40.9	9.97	0.02	4.08	40.9	9.97
ABF-Pre7-4C	0.00	5.95	100.3	5.93	0.00	5.94	100.6	5.91	0.00	5.94	100.7	5.90	0.00	5.94	100.8	5.90
AMG-J	0.09	2.38	89.2	2.67	0.13	2.05	89.5	2.29	0.13	2.00	89.5	2.24	0.14	2.00	89.6	2.23
AMG-4C	0.03	3.37	84.9	3.97	0.06	2.84	85.1	3.34	0.06	2.81	85.2	3.30	0.06	2.81	85.2	3.30
GMG-J	0.35	1.06	92.6	1.15	0.35	1.06	92.8	1.14	0.35	1.06	92.8	1.14	0.35	1.06	92.9	1.14
V(1,1)-4C	0.07	2.73	80.6	3.39	0.09	2.39	80.8	2.96	0.09	2.37	80.9	2.93	0.09	2.37	80.9	2.93
V(0,1)-4C	0.11	2.18	44.7	4.88	0.16	1.85	44.8	4.14	0.19	1.64	44.8	3.67	0.19	1.64	44.8	3.66
CMG	0.15	1.91	60.2	3.17	0.15	1.90	62.3	3.04	0.15	1.90	64.3	2.95	0.15	1.90	67.2	2.82
ABF-Pre7-W	0.00	11.63	195.6	5.94	0.00	11.63	236.1	4.92	0.00	11.63	281.7	4.13	0.00	11.63	339.1	3.43
ABF-Pre7-4C-W	0.00	11.97	620.6	1.93	0.00	11.97	784.2	1.53	0.00	11.97	960.3	1.25	0.00	11.97	1165.4	1.03
AMG-4C-W	0.02	4.16	132.6	3.14	0.02	4.16	137.1	3.04	0.02	4.16	139.6	2.98	0.02	4.16	141.1	2.95

(a) empirical convergence results

Algorithm	$L=4$						$L=5$						$L=6$						$L=7$					
	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	1231.24	82404.84	0.94	0.06	32.8	0.17	670.83	9573.08	0.93	0.08	32.7	0.24	283.32	89.89	0.89	0.12	32.7	0.36	253.00	20.31	0.88	0.13	32.7	0.38
ABF-Pre7	2.37	1606.59	0.21	1.55	40.9	3.79	2.37	860.71	0.21	1.55	40.9	3.79	2.37	856.90	0.21	1.55	40.9	3.79	2.37	621.09	0.21	1.55	40.9	3.78
ABF-Pre7-4C	1.17	1606.59	0.04	3.23	100.3	3.22	1.17	860.71	0.04	3.23	100.6	3.21	1.17	856.90	0.04	3.23	100.7	3.21	1.17	621.09	0.04	3.23	100.8	3.21
AMG-J	1.95	4789.26	0.17	1.80	89.2	2.02	2.54	2312.30	0.23	1.48	89.5	1.65	2.59	485.90	0.23	1.45	89.5	1.62	2.60	474.15	0.23	1.45	89.6	1.62
AMG-4C	1.58	4789.26	0.11	2.18	84.9	2.56	1.94	2312.30	0.16	1.81	85.1	2.13	1.96	485.90	0.17	1.79	85.2	2.10	1.96	474.15	0.17	1.79	85.2	2.10
GMG-J	6.69	82404.84	0.44	0.82	92.6	0.88	6.59	9573.08	0.44	0.82	92.8	0.89	6.34	89.89	0.43	0.84	92.8	0.91	6.22	20.31	0.43	0.85	92.9	0.91
V(1,1)-4C	1.58	4789.26	0.11	2.18	80.6	2.70	1.94	2312.30	0.16	1.81	80.8	2.24	1.96	485.90	0.17	1.79	80.9	2.21	1.96	474.15	0.17	1.79	80.9	2.21
V(0,1)-4C	1.69	4789.26	0.13	2.03	44.7	4.55	2.47	2312.30	0.22	1.50	44.8	3.36	3.07	485.90	0.27	1.30	44.8	2.90	3.11	474.15	0.28	1.29	44.8	2.87
CMG	2.26	100.82	0.20	1.60	60.2	2.66	2.27	35.09	0.20	1.60	62.3	2.56	2.27	12.43	0.20	1.60	64.3	2.48	2.27	1.00	0.20	1.60	67.2	2.38
ABF-Pre7-W	-	-	-	-	195.6	-	-	-	-	-	236.1	-	-	-	-	-	281.7	-	-	-	-	-	339.1	-
ABF-Pre7-4C-W	-	-	-	-	620.6	-	-	-	-	-	784.2	-	-	-	-	-	960.3	-	-	-	-	-	1165.4	-
AMG-4C-W	-	-	-	-	132.6	-	-	-	-	-	137.1	-	-	-	-	-	139.6	-	-	-	-	-	141.1	-

Original condition number =9378.4

(b) theoretical convergence results

Table 116: Colorization, 256×256 image

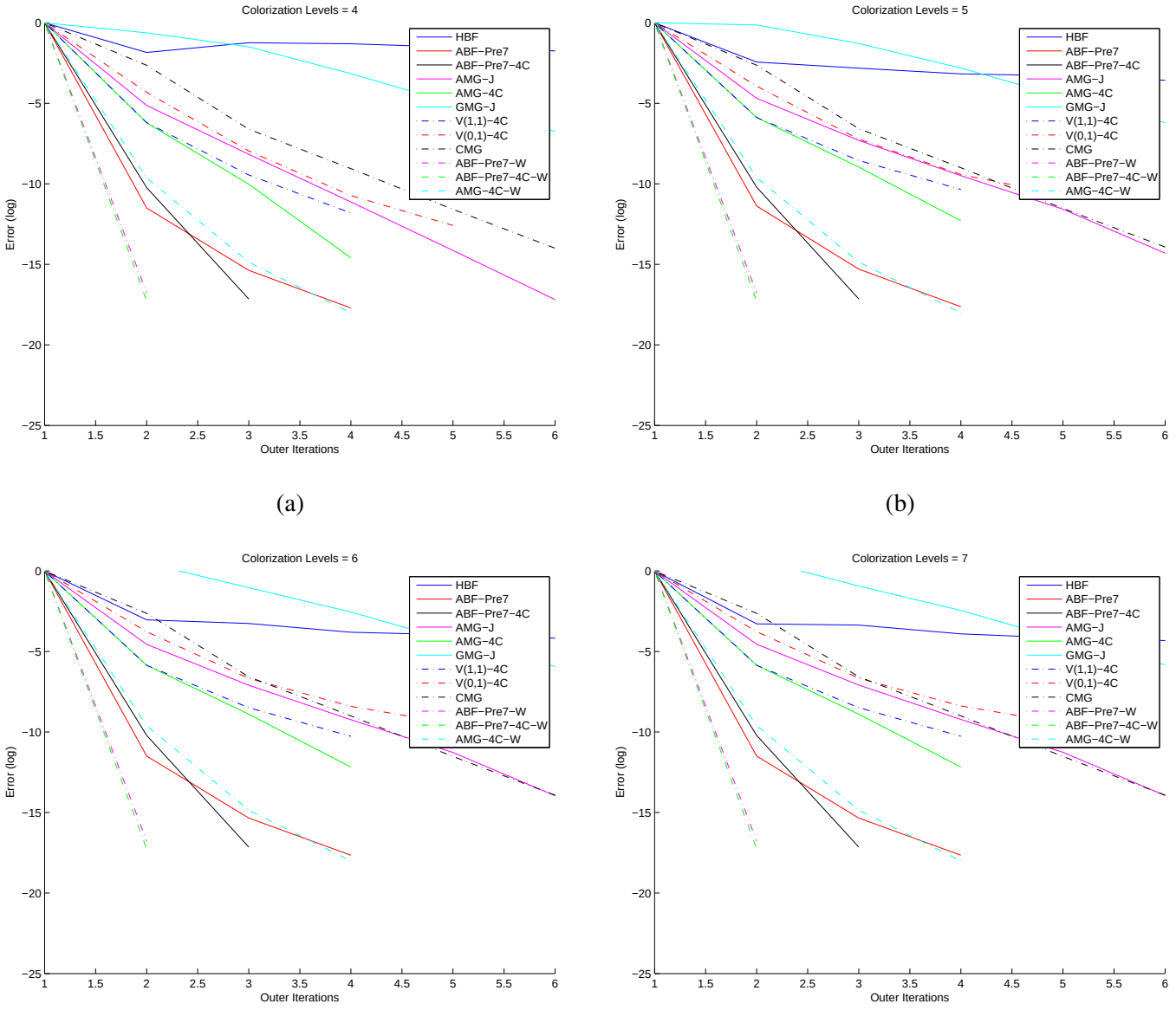


Figure 146: Log error vs. its, Colorization, 256×256 image

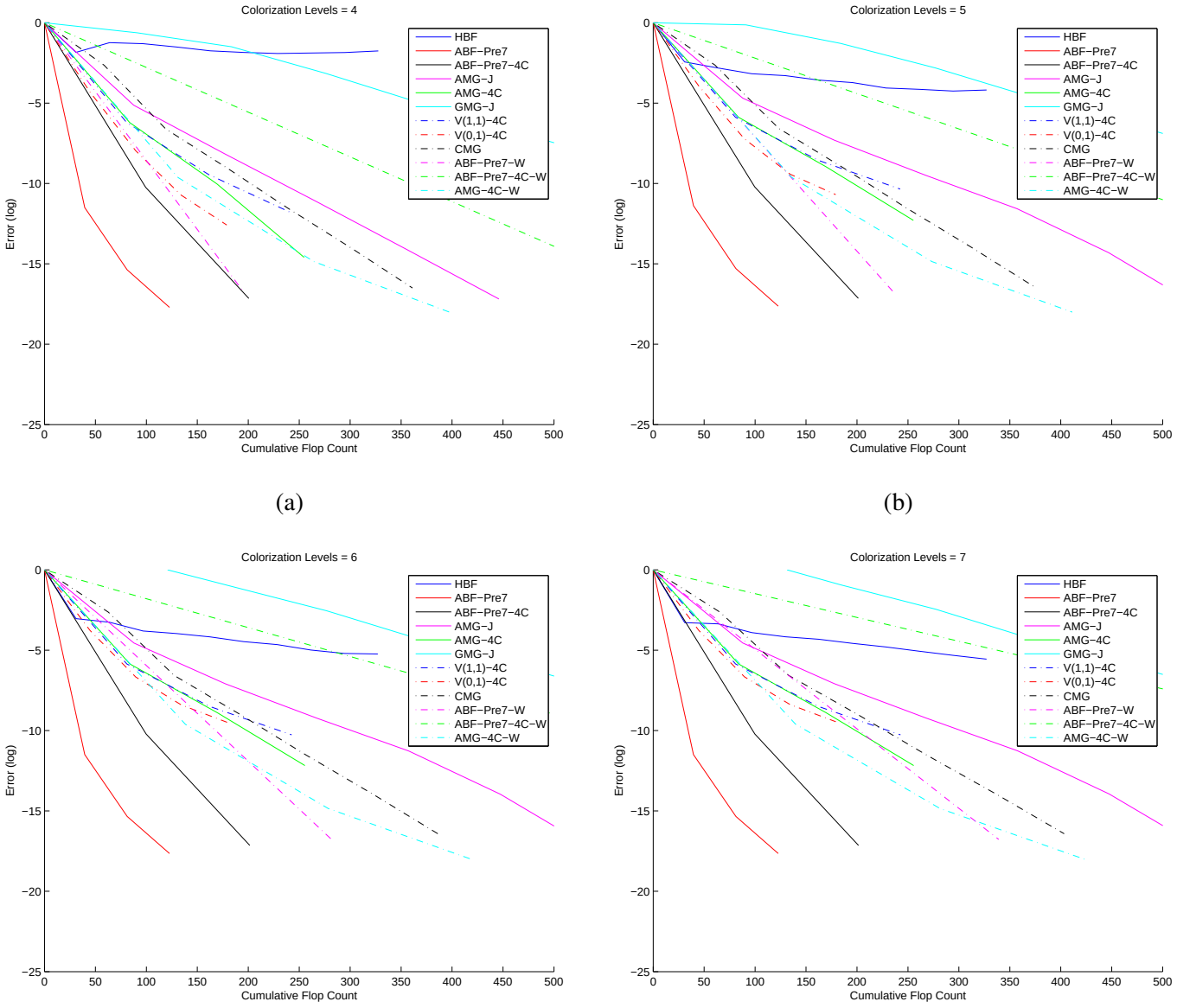


Figure 147: Log error vs. flops, Colorization, 256×256 image

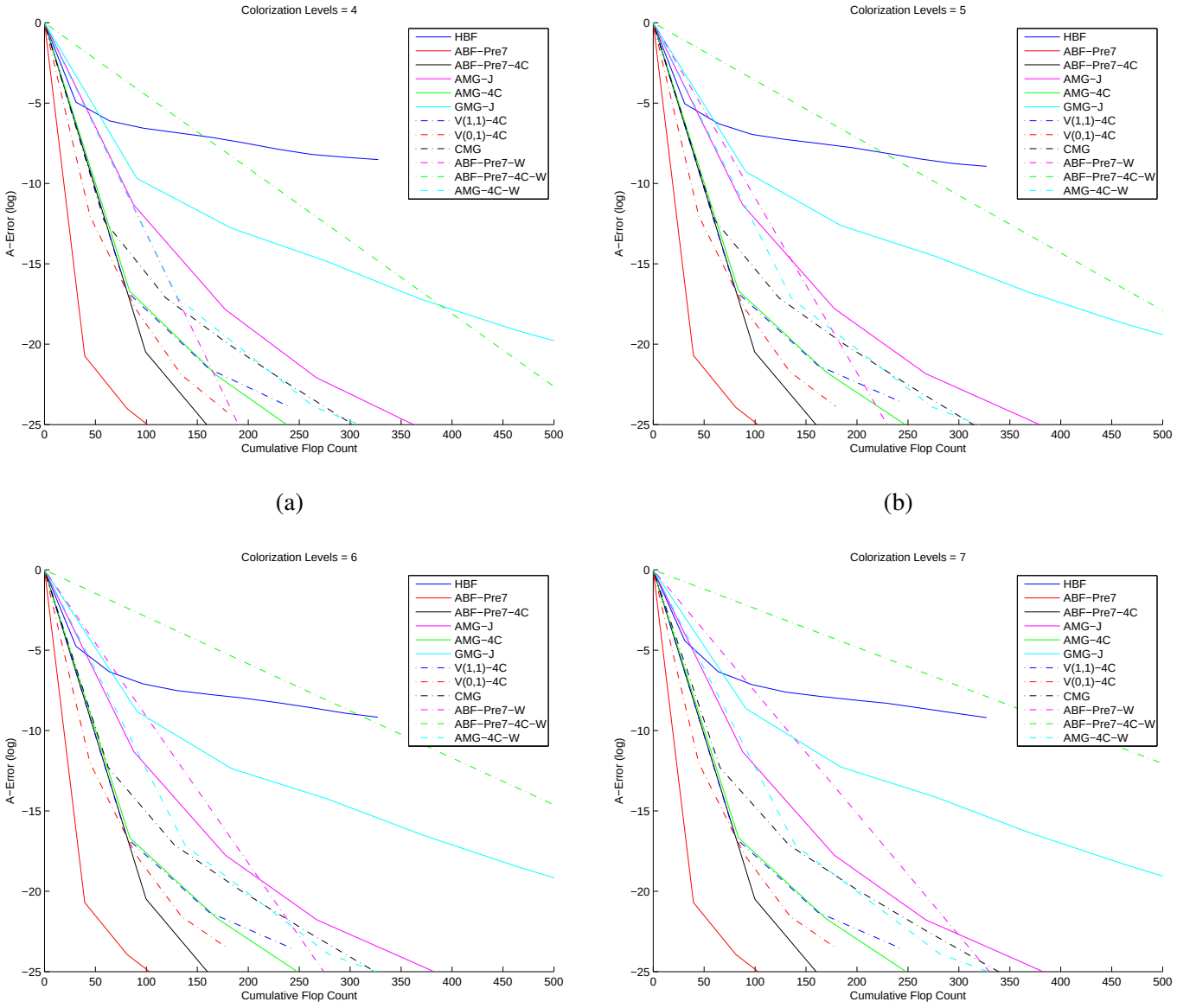


Figure 148: Log A-error vs. flops, Colorization, 256×256 image

	$L=3$				$L=4$				$L=5$				$L=6$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	1.01	-0.01	33.1	-0.02	0.88	0.12	32.8	0.38	0.88	0.13	32.7	0.40	0.74	0.31	32.7	0.93
ABF-Pre7	0.09	2.41	41.4	5.82	0.09	2.38	41.2	5.78	0.10	2.33	41.2	5.67	0.10	2.33	41.2	5.65
ABF-Pre7-4C	0.03	3.52	99.2	3.55	0.04	3.31	100.3	3.30	0.04	3.30	100.7	3.28	0.04	3.30	100.8	3.28
AMG-J	0.22	1.53	88.8	1.73	0.28	1.28	89.4	1.43	0.30	1.22	89.6	1.36	0.37	0.98	89.7	1.10
AMG-4C	0.13	2.07	84.6	2.44	0.17	1.79	85.1	2.10	0.20	1.59	85.3	1.87	0.24	1.43	85.4	1.68
GMG-J	0.45	0.81	92.0	0.88	0.47	0.75	92.6	0.81	0.48	0.73	92.8	0.79	0.49	0.72	92.9	0.78
V(1,1)-4C	0.21	1.58	80.1	1.97	0.25	1.40	80.6	1.73	0.28	1.27	80.8	1.57	0.32	1.15	80.9	1.43
V(0,1)-4C	0.26	1.34	44.8	3.00	0.30	1.19	44.7	2.67	0.35	1.06	44.8	2.36	0.32	1.13	44.8	2.52
CMG	0.18	1.73	61.4	2.82	0.19	1.68	64.6	2.61	0.19	1.66	67.2	2.47	0.19	1.66	69.4	2.39
ABF-Pre7-W	0.33	1.09	159.1	0.69	0.33	1.09	195.7	0.56	0.33	1.09	235.5	0.46	0.33	1.09	278.5	0.39
ABF-Pre7-4C-W	0.00	8.32	459.1	1.81	0.00	8.32	616.6	1.35	0.00	8.32	777.2	1.07	0.00	8.32	949.1	0.88
AMG-4C-W	0.06	2.76	124.3	2.22	0.07	2.60	132.5	1.96	0.08	2.52	137.0	1.84	0.08	2.52	139.4	1.81

(a) empirical convergence results

	$L=3$						$L=4$						$L=5$						$L=6$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	-	1029993.11	-	-	33.1	-	445.60	800891.29	0.91	0.09	32.8	0.29	565.70	326315.60	0.92	0.08	32.7	0.26	251.71	11691.29	0.88	0.13	32.7	0.39
ABF-Pre7	5.66	41358.37	0.41	0.90	41.4	2.16	6.82	10393.73	0.45	0.81	41.2	1.96	7.83	8210.36	0.47	0.75	41.2	1.82	8.01	4746.95	0.48	0.74	41.2	1.79
ABF-Pre7-4C	2.29	41358.37	0.20	1.59	99.2	1.60	2.51	10393.73	0.23	1.49	100.3	1.48	2.58	8210.36	0.23	1.46	100.7	1.45	2.58	4746.95	0.23	1.46	100.8	1.45
AMG-J	4.70	35972142.08	0.37	1.00	88.8	1.12	6.88	92909.29	0.45	0.80	89.4	0.90	6.94	1655.10	0.45	0.80	89.6	0.89	7.42	289.63	0.46	0.77	89.7	0.86
AMG-4C	4.41	35972142.08	0.35	1.04	84.6	1.23	6.80	92909.29	0.45	0.81	85.1	0.95	7.37	1655.10	0.46	0.77	85.3	0.91	7.62	289.63	0.47	0.76	85.4	0.89
GMG-J	7.76	1029993.11	0.47	0.75	92.0	0.82	8.50	800891.29	0.49	0.71	92.6	0.77	9.43	326315.60	0.51	0.68	92.8	0.73	9.55	11691.29	0.51	0.67	92.9	0.72
V(1,1)-4C	4.41	35972142.08	0.35	1.04	80.1	1.29	6.80	92909.29	0.45	0.81	80.6	1.00	7.37	1655.10	0.46	0.77	80.8	0.96	7.62	289.63	0.47	0.76	80.9	0.94
V(0,1)-4C	6.47	35972142.08	0.44	0.83	44.8	1.85	11.13	92909.29	0.54	0.62	44.7	1.38	15.23	1655.10	0.59	0.52	44.8	1.17	14.47	289.63	0.58	0.54	44.8	1.20
CMG	2.41	1829.90	0.22	1.53	61.4	2.50	2.44	487.70	0.22	1.52	64.6	2.35	2.45	186.09	0.22	1.51	67.2	2.25	2.45	87.86	0.22	1.51	69.4	2.18
ABF-Pre7-W	-	-	-	-	159.1	-	-	-	-	-	195.7	-	-	-	-	-	235.5	-	-	-	-	-	278.5	-
ABF-Pre7-4C-W	-	-	-	-	459.1	-	-	-	-	-	616.6	-	-	-	-	-	777.2	-	-	-	-	-	949.1	-
AMG-4C-W	-	-	-	-	124.3	-	-	-	-	-	132.5	-	-	-	-	-	137.0	-	-	-	-	-	139.4	-

Original condition number =16698.2

(b) theoretical convergence results

Table 117: Colorization, 239×319 image

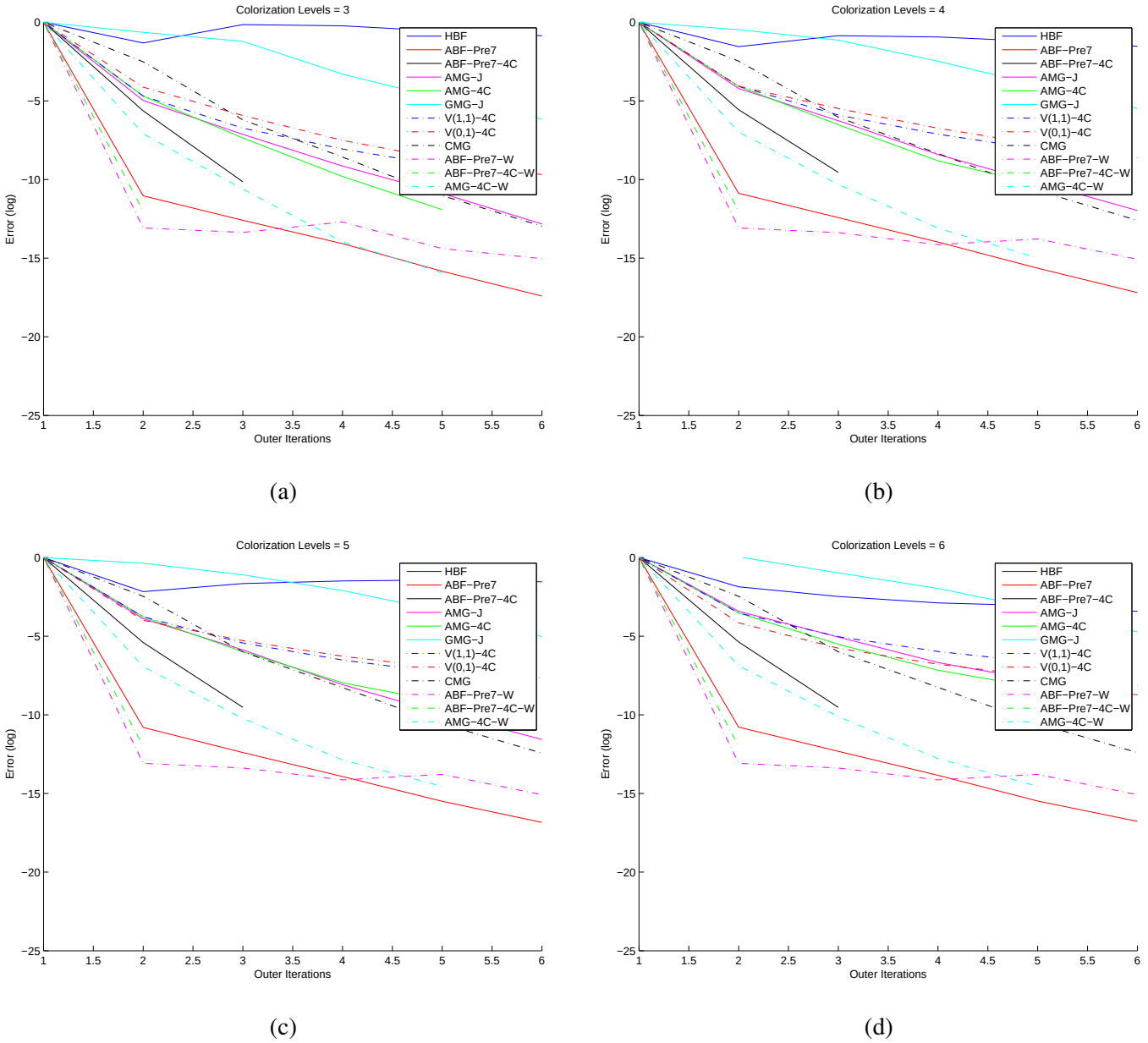
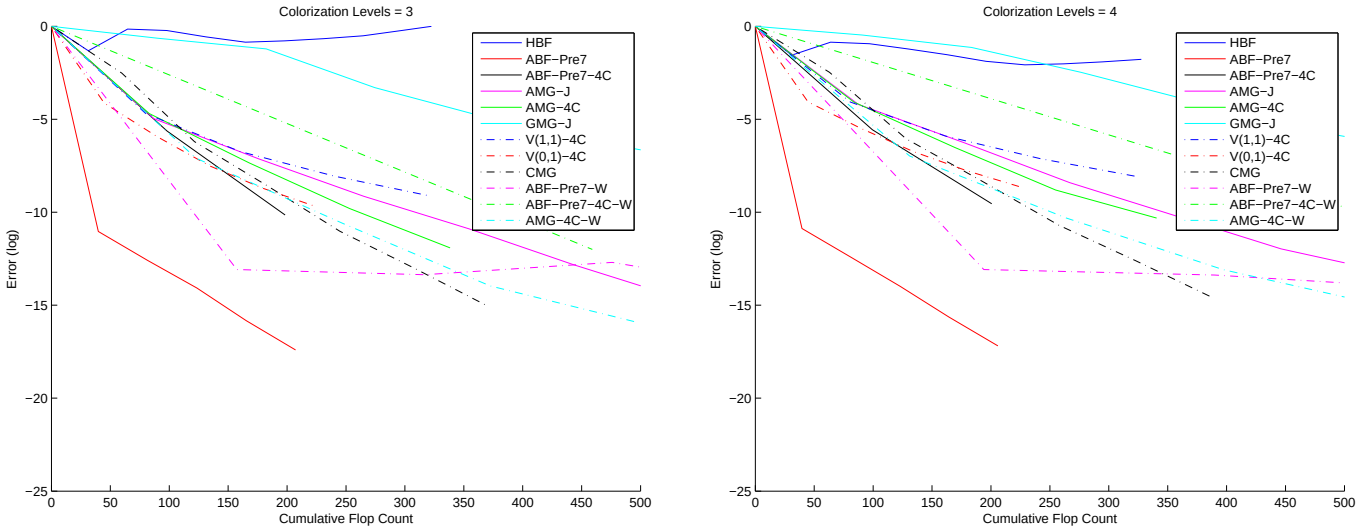
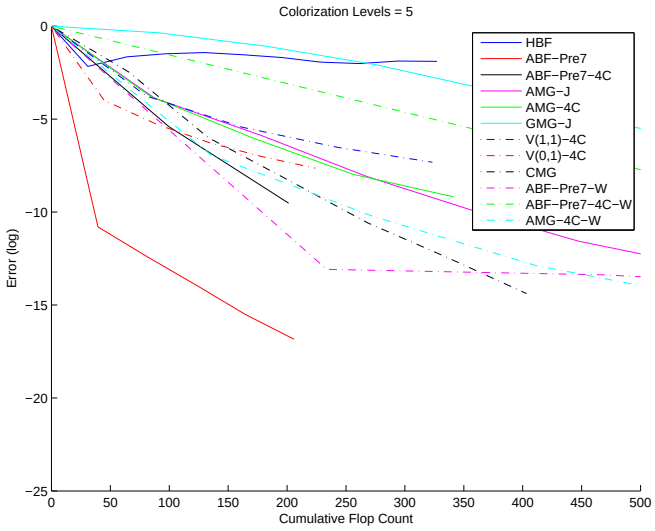


Figure 149: Log error vs. its, Colorization, 239×319 image

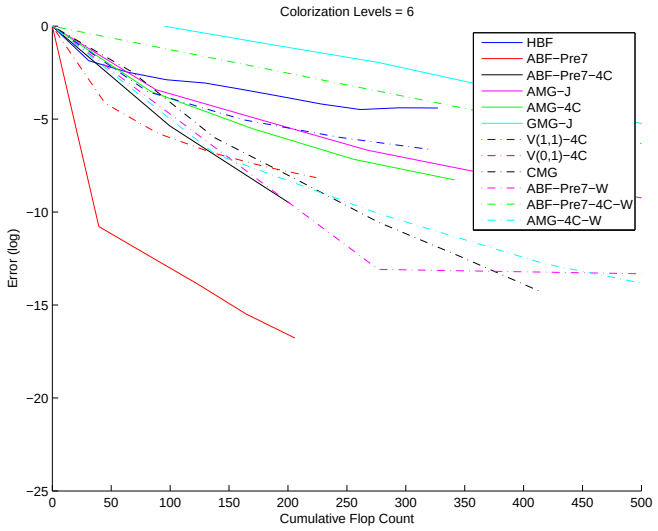


(a)

(b)



(c)



(d)

Figure 150: Log error vs. flops, Colorization, 239×319 image

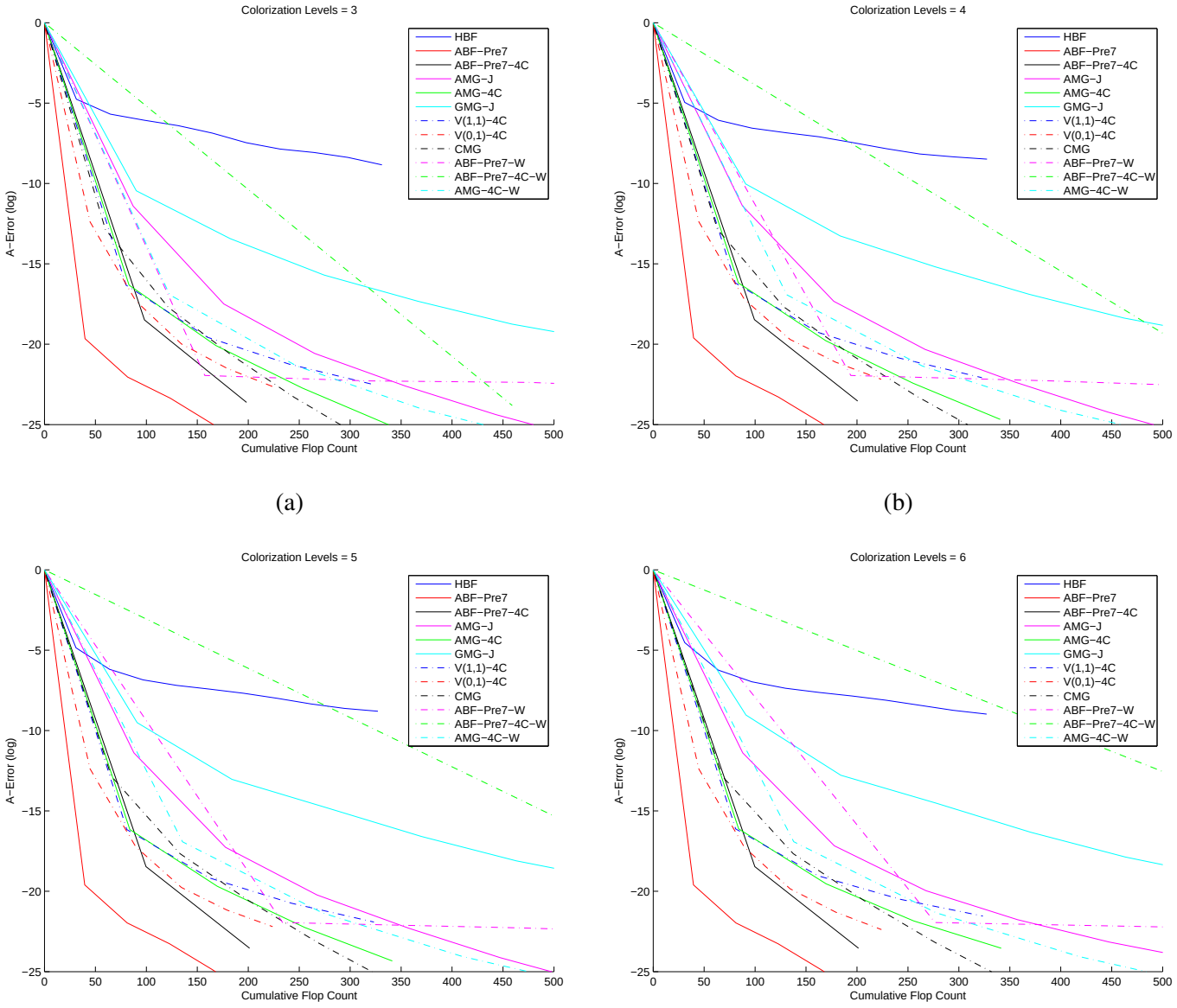


Figure 151: Log A-error vs. flops, Colorization, 239×319 image

	$L=5$				$L=6$				$L=7$				$L=8$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.86	0.15	32.7	0.47	0.97	0.04	32.6	0.11	0.98	0.02	32.6	0.07	0.87	0.13	32.6	0.41
ABF-Pre7	0.00	8.43	40.5	20.79	0.00	8.36	40.5	20.65	0.00	8.28	40.5	20.45	0.00	8.24	40.5	20.35
ABF-Pre7-4C	0.00	12.44	99.4	12.52	0.00	12.30	99.5	12.36	0.00	12.16	99.5	12.22	0.00	12.11	99.5	12.17
AMG-J	0.03	3.66	89.0	4.12	0.03	3.50	89.0	3.94	0.04	3.26	89.0	3.66	0.07	2.67	89.0	3.00
AMG-4C	0.00	8.65	83.6	10.34	0.00	8.25	83.7	9.86	0.00	7.89	83.7	9.42	0.00	7.21	83.7	8.62
GMG-J	0.28	1.27	92.5	1.37	0.31	1.18	92.6	1.27	0.35	1.06	92.6	1.15	0.38	0.97	92.6	1.05
V(1,1)-4C	0.00	8.65	80.6	10.72	0.00	8.25	80.7	10.23	0.00	7.89	80.7	9.77	0.00	7.21	80.7	8.94
V(0,1)-4C	0.01	4.61	44.7	10.33	0.02	3.91	44.7	8.74	0.03	3.51	44.7	7.86	0.53	0.63	44.7	1.41
CMG	0.03	3.38	57.9	5.84	0.04	3.32	58.4	5.69	0.04	3.31	58.8	5.62	0.04	3.30	59.2	5.58

(a) empirical convergence results

Table 118: Colorization, 1854×2048 image

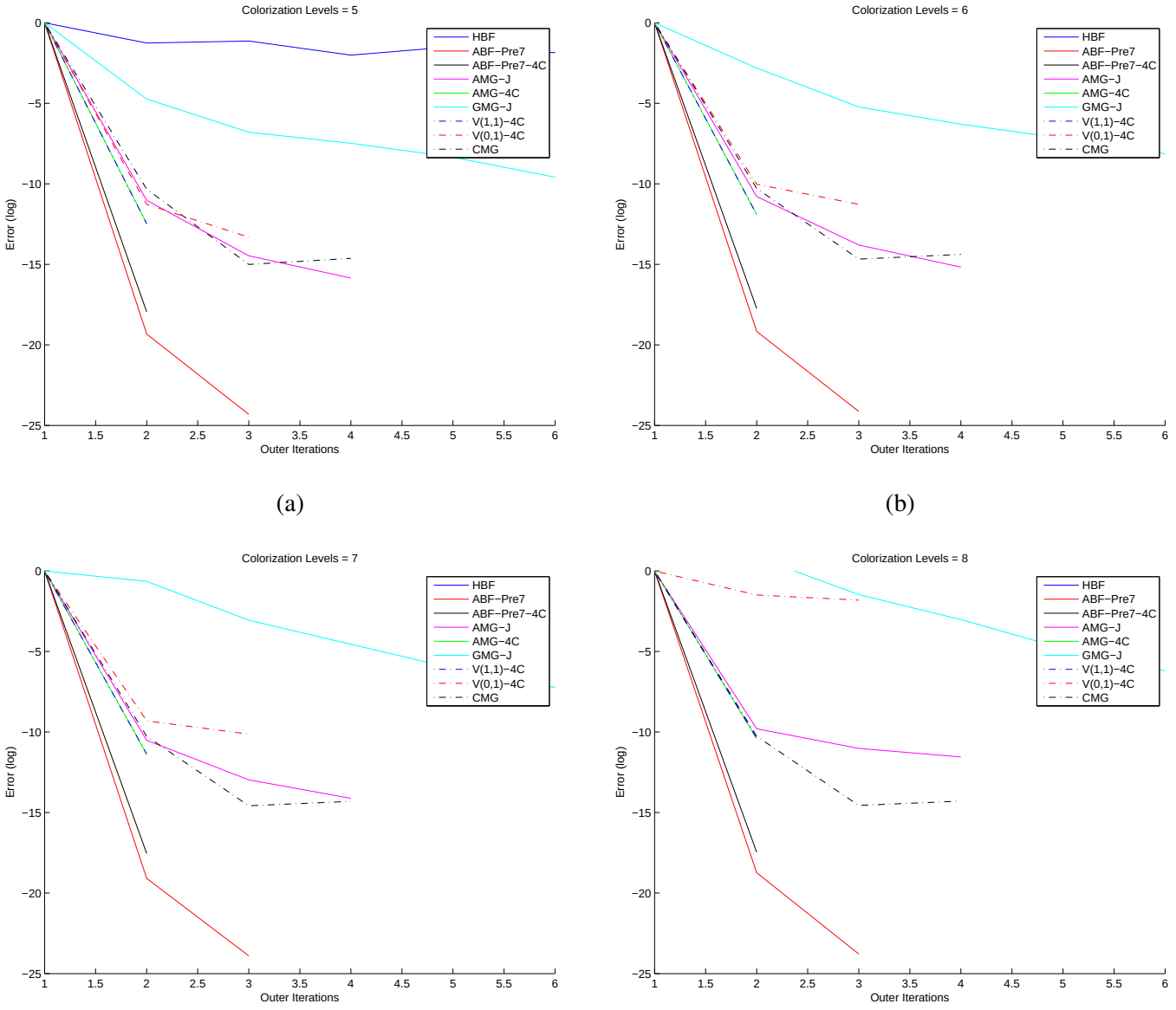
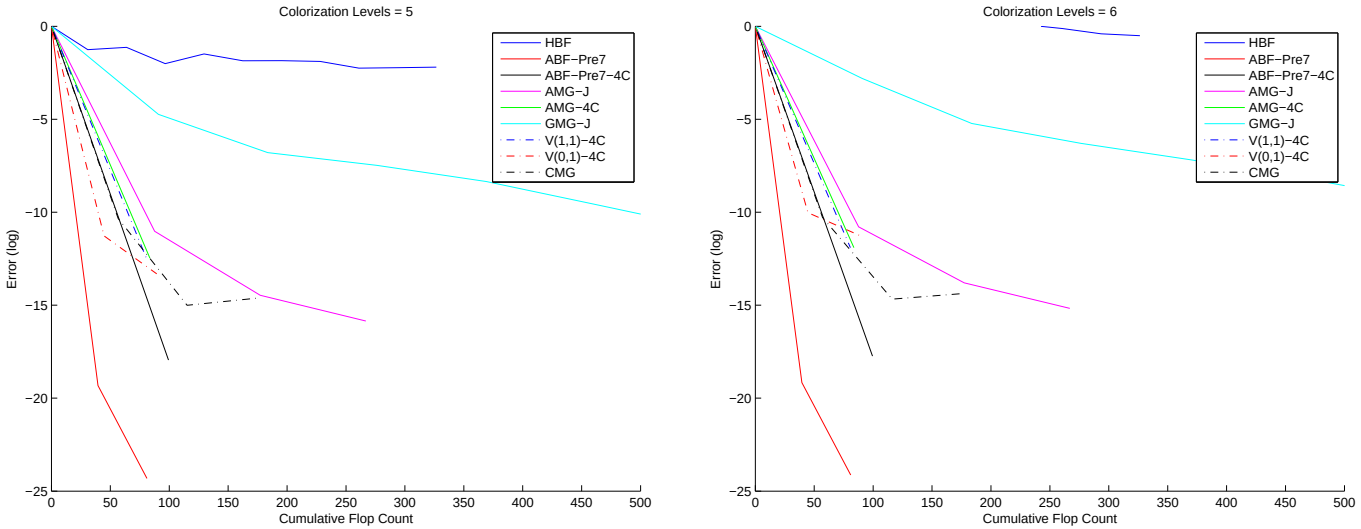
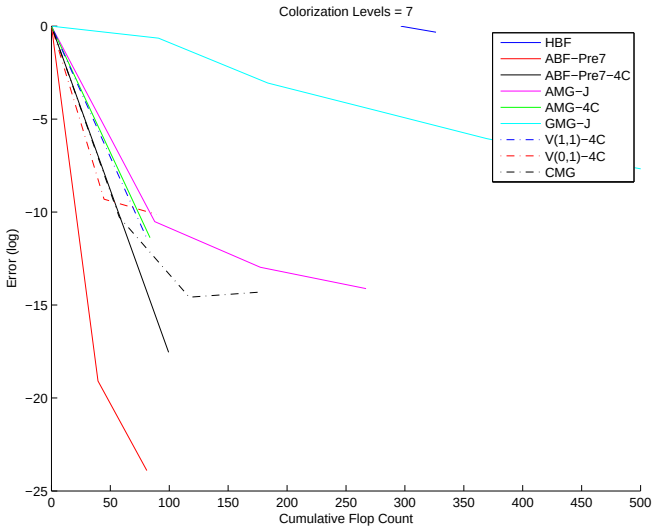


Figure 152: Log error vs. its, Colorization, 1854×2048 image

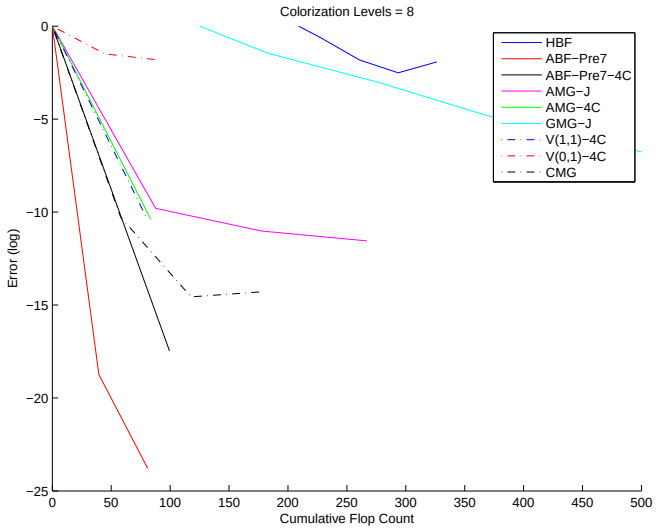


(a)

(b)

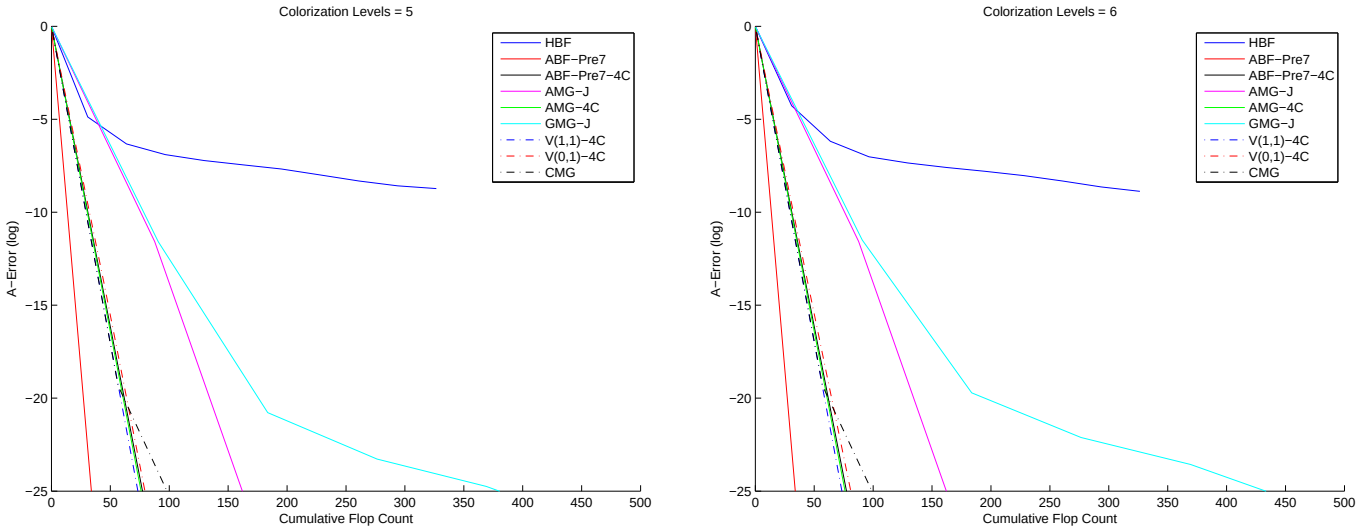


(c)



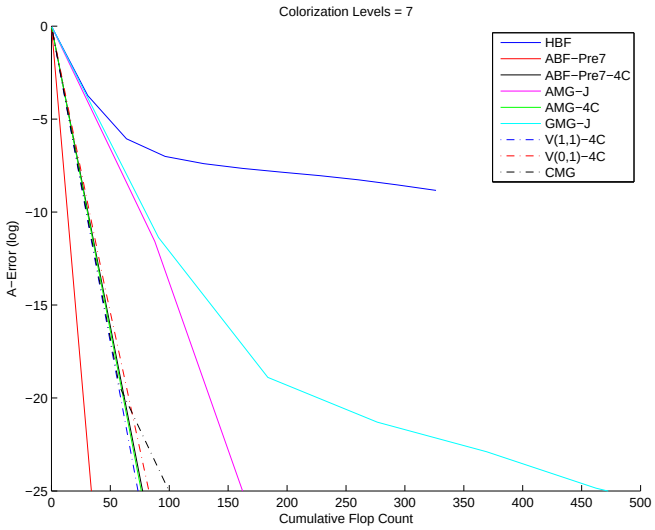
(d)

Figure 153: Log error vs. flops, Colorization, 1854×2048 image

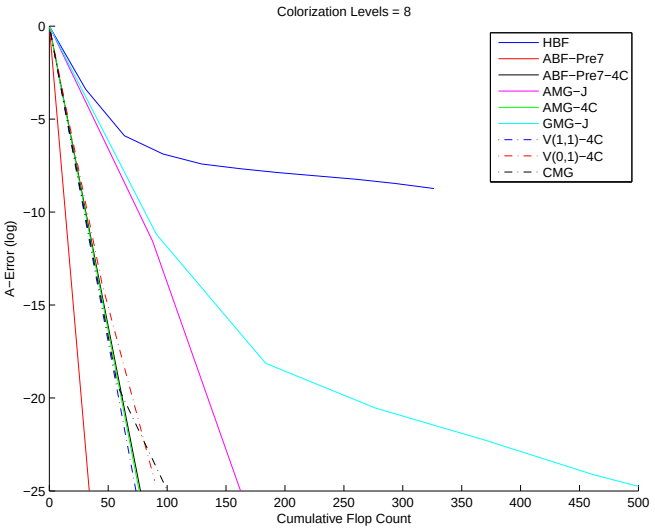


(a)

(b)



(c)



(d)

Figure 154: Log A-error vs. flops, Colorization, 1854×2048 image

	$L=1$				$L=2$				$L=3$				$L=4$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.77	0.26	43.1	0.61	0.73	0.32	40.6	0.78	0.67	0.41	40.5	1.00	0.68	0.39	40.6	0.96
ABF-Pre7	0.10	2.32	49.5	4.69	0.11	2.24	48.6	4.61	0.11	2.24	49.0	4.57	0.11	2.24	49.1	4.55
ABF-Pre7-4C	0.01	4.63	139.1	3.33	0.01	4.46	160.8	2.78	0.01	4.46	166.7	2.68	0.01	4.46	168.3	2.65
AMG-J	0.02	4.05	103.4	3.92	0.02	3.90	116.1	3.36	0.02	3.79	119.8	3.17	0.02	3.78	120.8	3.13
AMG-4C	0.00	5.52	100.2	5.50	0.01	4.66	112.2	4.15	0.01	4.69	115.7	4.05	0.01	4.69	116.7	4.02
GMG-J	0.09	2.38	106.0	2.25	0.09	2.37	119.3	1.99	0.10	2.35	123.1	1.91	0.10	2.32	124.2	1.87
V(1,1)-4C	0.01	5.10	95.9	5.32	0.01	4.43	107.8	4.11	0.01	4.46	111.4	4.01	0.01	4.46	112.3	3.97
V(0,1)-4C	0.05	2.95	56.9	5.19	0.05	3.08	59.1	5.21	0.05	3.08	60.2	5.12	0.05	3.08	60.5	5.09
CMG	0.04	3.24	86.1	3.76	0.04	3.24	128.0	2.53	0.04	3.24	164.8	1.97	0.04	3.24	207.8	1.56
ABF-Pre7-W	0.10	2.34	71.5	3.27	0.10	2.34	109.9	2.13	0.10	2.34	150.0	1.56	0.10	2.34	191.7	1.22
ABF-Pre7-4C-W	0.01	4.77	191.2	2.49	0.01	4.77	349.5	1.36	0.01	4.77	509.6	0.94	0.01	4.77	671.3	0.71
AMG-4C-W	0.00	5.52	100.2	5.50	0.00	5.45	135.2	4.03	0.00	5.45	153.6	3.55	0.00	5.45	163.1	3.34

(a) empirical convergence results

	$L=1$						$L=2$						$L=3$						$L=4$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	4.73	2583.17	0.37	0.99	43.1	2.31	15.01	424.19	0.59	0.53	40.6	1.30	17.97	58.29	0.62	0.48	40.5	1.19	23.63	35.92	0.66	0.42	40.6	1.03
ABF-Pre7	2.31	2105.41	0.21	1.58	49.5	3.18	2.36	873.17	0.21	1.55	48.6	3.19	2.37	432.36	0.21	1.55	49.0	3.16	2.37	268.57	0.21	1.55	49.1	3.15
ABF-Pre7-4C	1.45	2105.41	0.09	2.38	139.1	1.71	1.54	873.17	0.11	2.23	160.8	1.39	1.54	432.36	0.11	2.23	166.7	1.34	1.54	268.57	0.11	2.23	168.3	1.33
AMG-J	1.39	1413.95	0.08	2.51	103.4	2.42	1.40	349.70	0.08	2.48	116.1	2.14	1.40	213.75	0.08	2.47	119.8	2.06	1.41	212.86	0.08	2.47	120.8	2.04
AMG-4C	1.09	1413.95	0.02	3.81	100.2	3.80	1.11	349.70	0.03	3.65	112.2	3.26	1.11	213.75	0.03	3.67	115.7	3.18	1.11	212.86	0.03	3.67	116.7	3.15
GMG-J	2.57	2583.17	0.23	1.46	106.0	1.38	3.71	424.19	0.32	1.15	119.3	0.96	3.78	58.29	0.32	1.14	123.1	0.92	3.77	35.92	0.32	1.14	124.2	0.92
V(1,1)-4C	1.09	1413.95	0.02	3.81	95.9	3.97	1.11	349.70	0.03	3.65	107.8	3.39	1.11	213.75	0.03	3.67	111.4	3.30	1.11	212.86	0.03	3.67	112.3	3.27
V(0,1)-4C	1.31	1413.95	0.07	2.70	56.9	4.74	1.75	349.70	0.14	1.98	59.1	3.35	1.55	213.75	0.11	2.21	60.2	3.68	1.55	212.86	0.11	2.21	60.5	3.66
CMG	1.61	139.63	0.12	2.13	86.1	2.47	1.61	56.44	0.12	2.13	128.0	1.66	1.61	14.12	0.12	2.13	164.8	1.29	1.61	1.00	0.12	2.13	207.8	1.03
ABF-Pre7-W	2.36	2105.41	0.21	1.55	71.5	2.17	2.36	873.17	0.21	1.55	109.9	1.41	2.36	432.36	0.21	1.55	150.0	1.03	2.36	268.57	0.21	1.55	191.7	0.81
ABF-Pre7-4C-W	1.41	2105.41	0.09	2.46	191.2	1.29	1.41	873.17	0.09	2.46	349.5	0.70	1.41	432.36	0.09	2.46	509.6	0.48	1.41	268.57	0.09	2.46	671.3	0.37
AMG-4C-W	1.09	1413.95	0.02	3.81	100.2	3.80	1.09	349.70	0.02	3.80	135.2	2.81	1.09	213.75	0.02	3.80	153.6	2.47	1.09	212.86	0.02	3.80	163.1	2.33

Original condition number =497.7

(b) theoretical convergence results

Table 119: Colorization 9-pt, 32×32 image

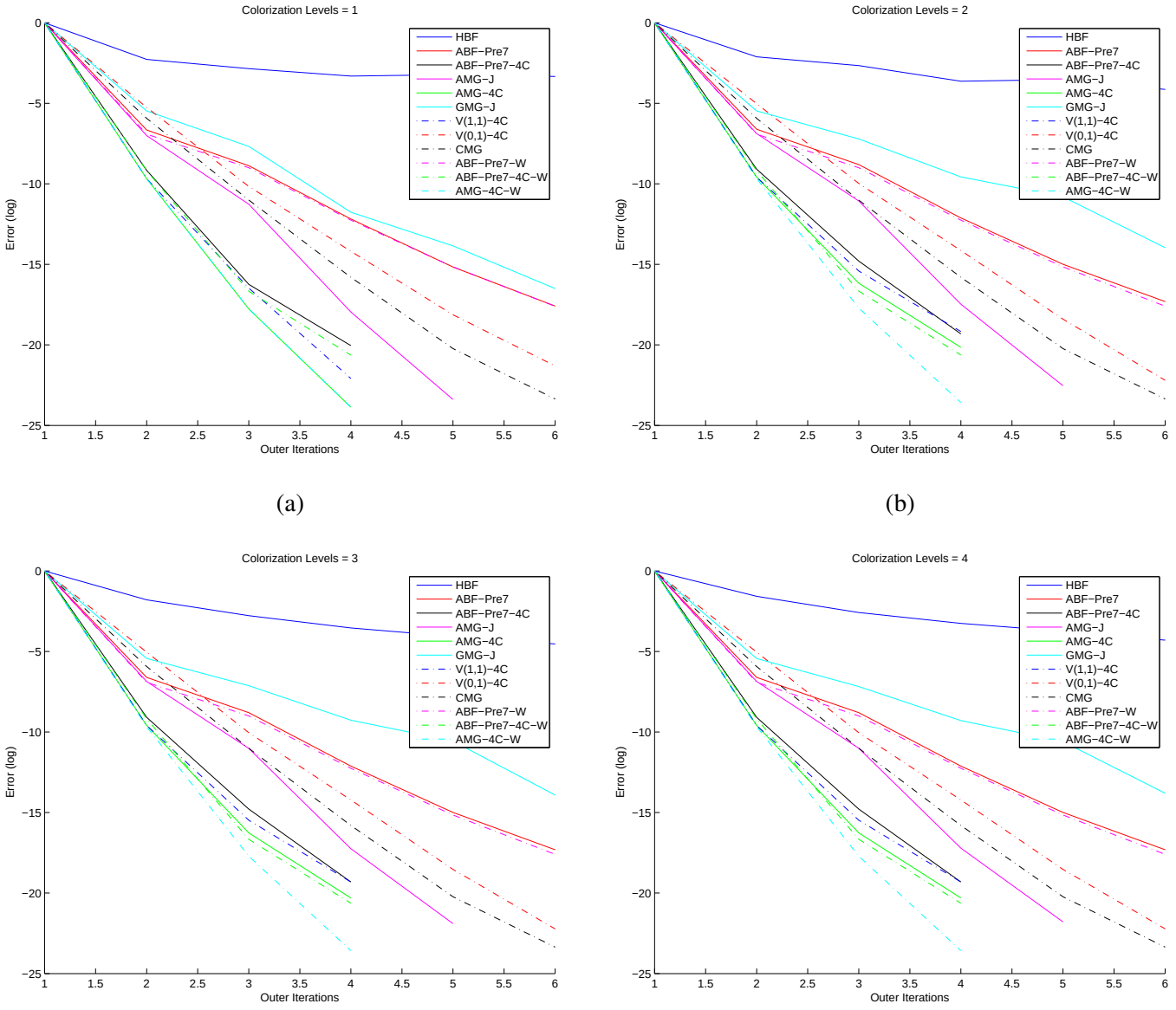
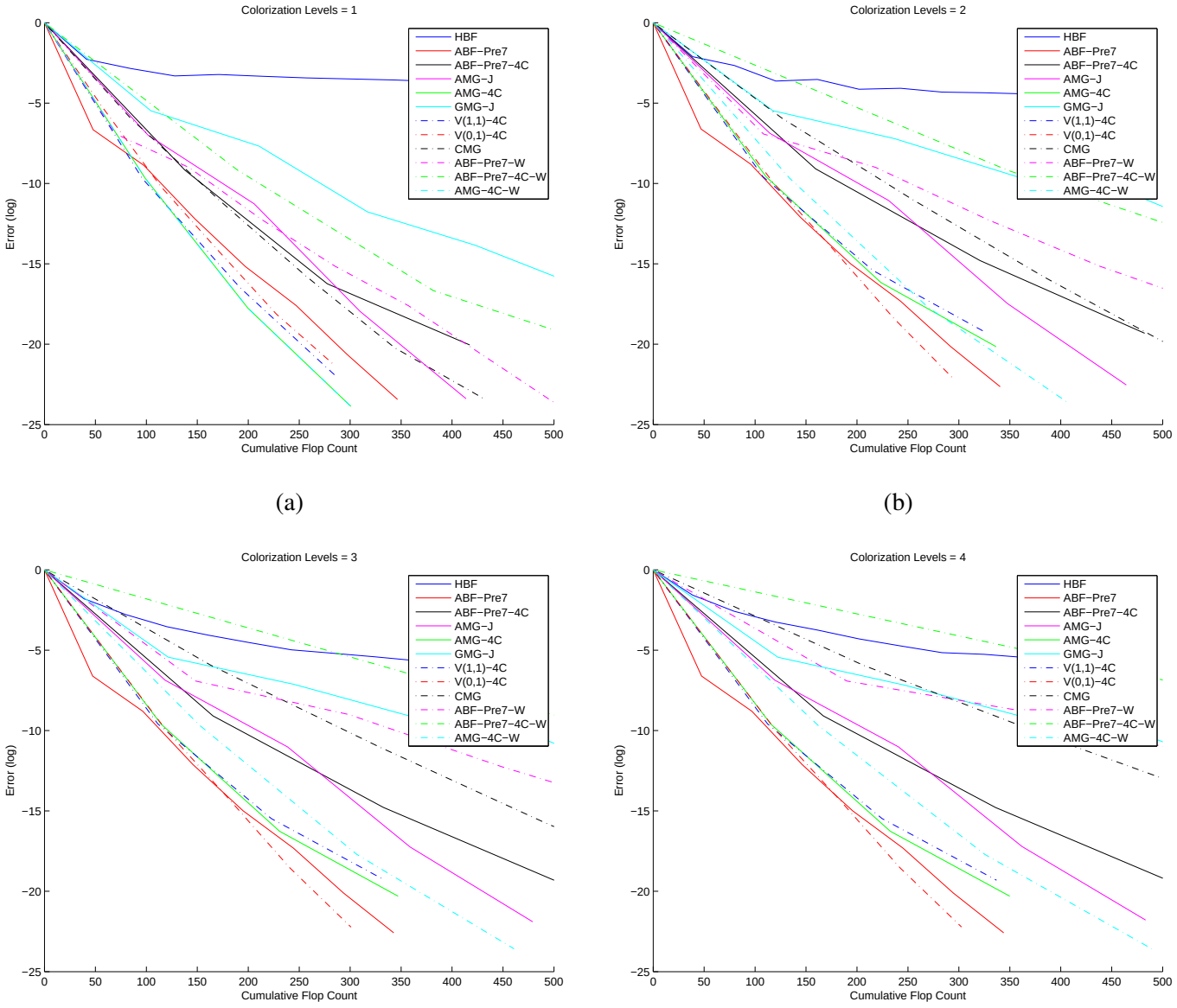


Figure 155: Log error vs. its, Colorization 9-pt, 32×32 image



(a) (b)

(c) (d)

Figure 156: Log error vs. flops, Colorization 9-pt, 32×32 image

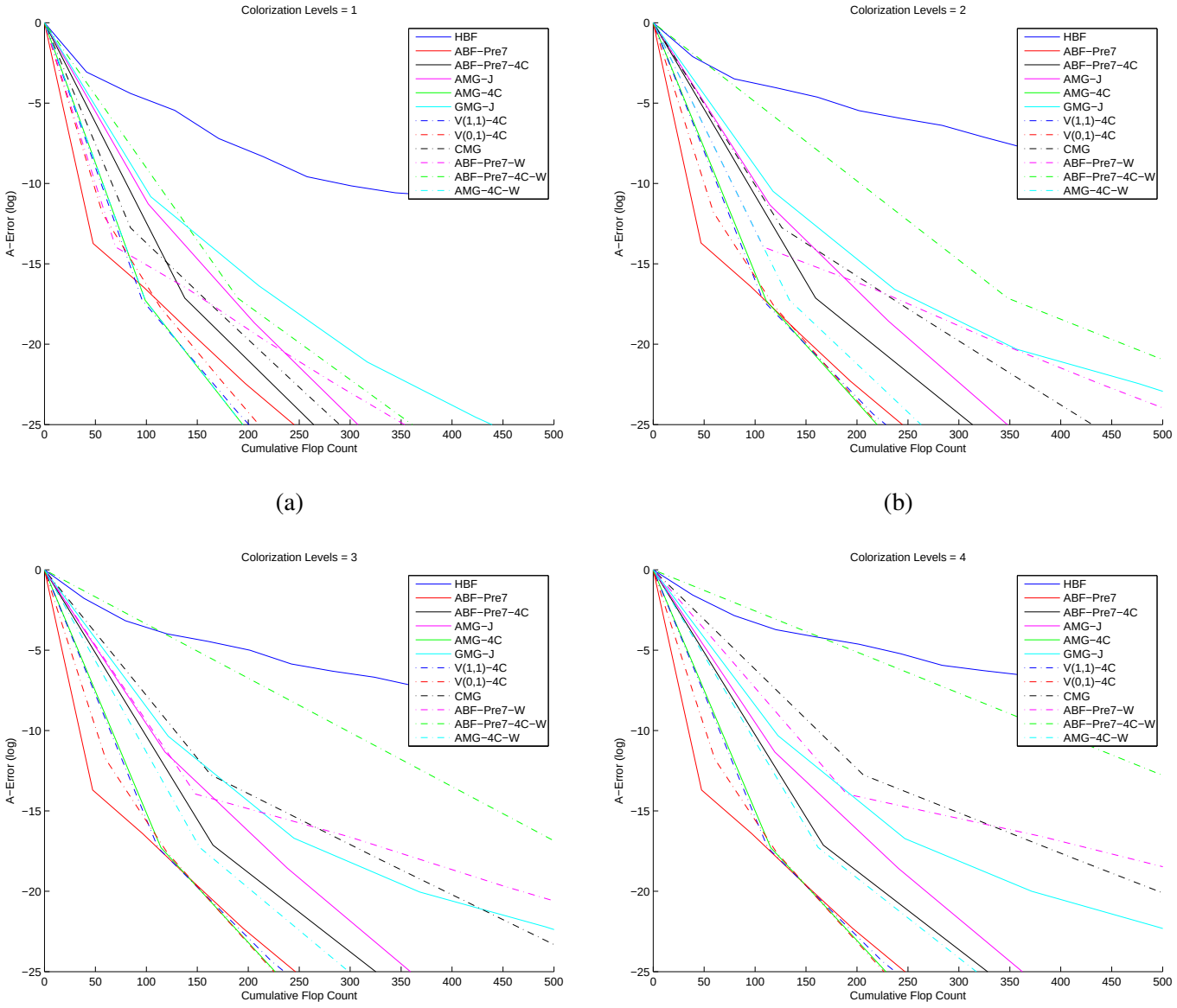


Figure 157: Log A-error vs. flops, Colorization 9-pt, 32×32 image

	$L=1$				$L=2$				$L=3$				$L=4$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.80	0.23	43.1	0.52	0.77	0.26	40.6	0.64	0.74	0.30	40.5	0.75	0.72	0.33	40.6	0.80
ABF-Pre7	0.16	1.83	49.5	3.69	0.16	1.83	48.6	3.76	0.16	1.83	49.0	3.74	0.16	1.83	49.1	3.72
ABF-Pre7-4C	0.00	6.01	138.8	4.33	0.00	6.05	160.4	3.77	0.00	6.05	166.4	3.64	0.00	6.05	168.0	3.60
AMG-J	0.05	3.04	103.4	2.94	0.05	2.98	116.2	2.56	0.05	2.95	119.9	2.46	0.05	2.97	120.9	2.45
AMG-4C	0.00	5.57	99.9	5.57	0.00	5.72	111.8	5.11	0.00	5.63	115.4	4.88	0.00	5.67	116.3	4.87
GMG-J	0.15	1.91	106.0	1.80	0.17	1.78	119.3	1.49	0.17	1.77	123.1	1.44	0.16	1.85	124.2	1.49
V(1,1)-4C	0.00	5.52	95.9	5.76	0.00	5.67	107.8	5.26	0.00	5.58	111.4	5.01	0.00	5.61	112.3	5.00
V(0,1)-4C	0.19	1.66	56.9	2.91	0.26	1.34	59.1	2.27	0.26	1.34	60.2	2.23	0.26	1.34	60.5	2.22
ABF-Pre7-W	0.26	1.35	71.6	1.88	0.26	1.35	109.9	1.23	0.26	1.35	150.1	0.90	0.26	1.35	191.8	0.70
ABF-Pre7-4C-W	0.00	6.15	190.8	3.22	0.00	6.15	349.1	1.76	0.00	6.15	509.3	1.21	0.00	6.15	671.0	0.92
AMG-4C-W	0.00	5.57	99.9	5.57	0.00	5.57	134.9	4.13	0.00	5.57	153.3	3.63	0.00	5.57	162.8	3.42

(a) empirical convergence results

	$L=1$						$L=2$						$L=3$						$L=4$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	5.84	13359.95	0.41	0.88	43.1	2.04	16.17	13864.29	0.60	0.51	40.6	1.25	19.92	355.56	0.63	0.46	40.5	1.12	26.32	40.61	0.67	0.39	40.6	0.97
ABF-Pre7	4.23	6549.70	0.35	1.06	49.5	2.15	4.42	5620.32	0.36	1.03	48.6	2.13	4.76	3993.06	0.37	0.99	49.0	2.02	4.76	3931.87	0.37	0.99	49.1	2.01
ABF-Pre7-4C	1.31	6549.70	0.07	2.69	138.8	1.93	1.31	5620.32	0.07	2.71	160.4	1.69	1.30	3993.06	0.07	2.71	166.4	1.63	1.30	3931.87	0.07	2.71	168.0	1.62
AMG-J	2.35	4597.45	0.21	1.56	103.4	1.51	2.34	4537.03	0.21	1.56	116.2	1.34	2.38	932.27	0.21	1.54	119.9	1.29	2.42	842.27	0.22	1.53	120.9	1.26
AMG-4C	1.72	4597.45	0.14	2.00	99.9	2.00	1.82	4537.03	0.15	1.91	111.8	1.71	1.82	932.27	0.15	1.90	115.4	1.65	1.82	842.27	0.15	1.90	116.3	1.64
GMG-J	5.55	13359.95	0.40	0.91	106.0	0.86	5.01	13864.29	0.38	0.96	119.3	0.81	4.85	355.56	0.38	0.98	123.1	0.80	4.76	40.61	0.37	0.99	124.2	0.80
V(1,1)-4C	1.72	4597.45	0.14	2.00	95.9	2.09	1.82	4537.03	0.15	1.91	107.8	1.77	1.82	932.27	0.15	1.90	111.4	1.71	1.82	842.27	0.15	1.90	112.3	1.69
V(0,1)-4C	2.68	4597.45	0.24	1.42	56.9	2.49	3.18	4537.03	0.28	1.27	59.1	2.15	3.18	932.27	0.28	1.27	60.2	2.11	3.17	842.27	0.28	1.27	60.5	2.10
ABF-Pre7-W	9.82	6549.70	0.52	0.66	71.6	0.92	9.82	5620.32	0.52	0.66	109.9	0.60	9.82	3993.06	0.52	0.66	150.1	0.44	9.82	3931.87	0.52	0.66	191.8	0.34
ABF-Pre7-4C-W	1.29	6549.70	0.06	2.76	190.8	1.45	1.29	5620.32	0.06	2.76	349.1	0.79	1.29	3993.06	0.06	2.76	509.3	0.54	1.29	3931.87	0.06	2.76	671.0	0.41
AMG-4C-W	1.72	4597.45	0.14	2.00	99.9	2.00	1.72	4537.03	0.13	2.00	134.9	1.49	1.72	932.27	0.13	2.00	153.3	1.31	1.72	842.27	0.13	2.00	162.8	1.23

Original condition number =1821.6

(b) theoretical convergence results

Table 120: Colorization 9-pt, 32×32 image

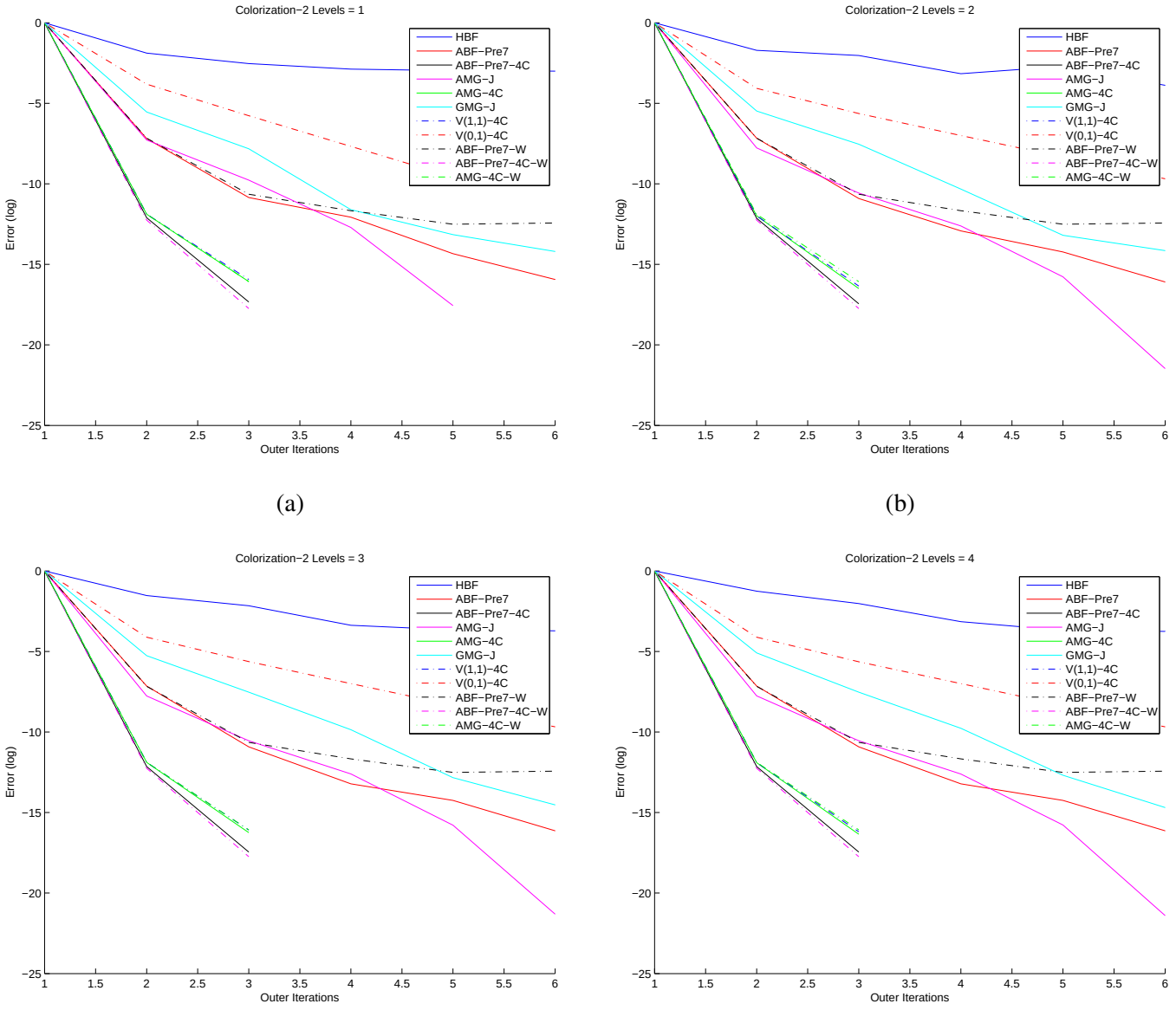


Figure 158: Log error vs. its, Colorization 9-pt, 32×32 image

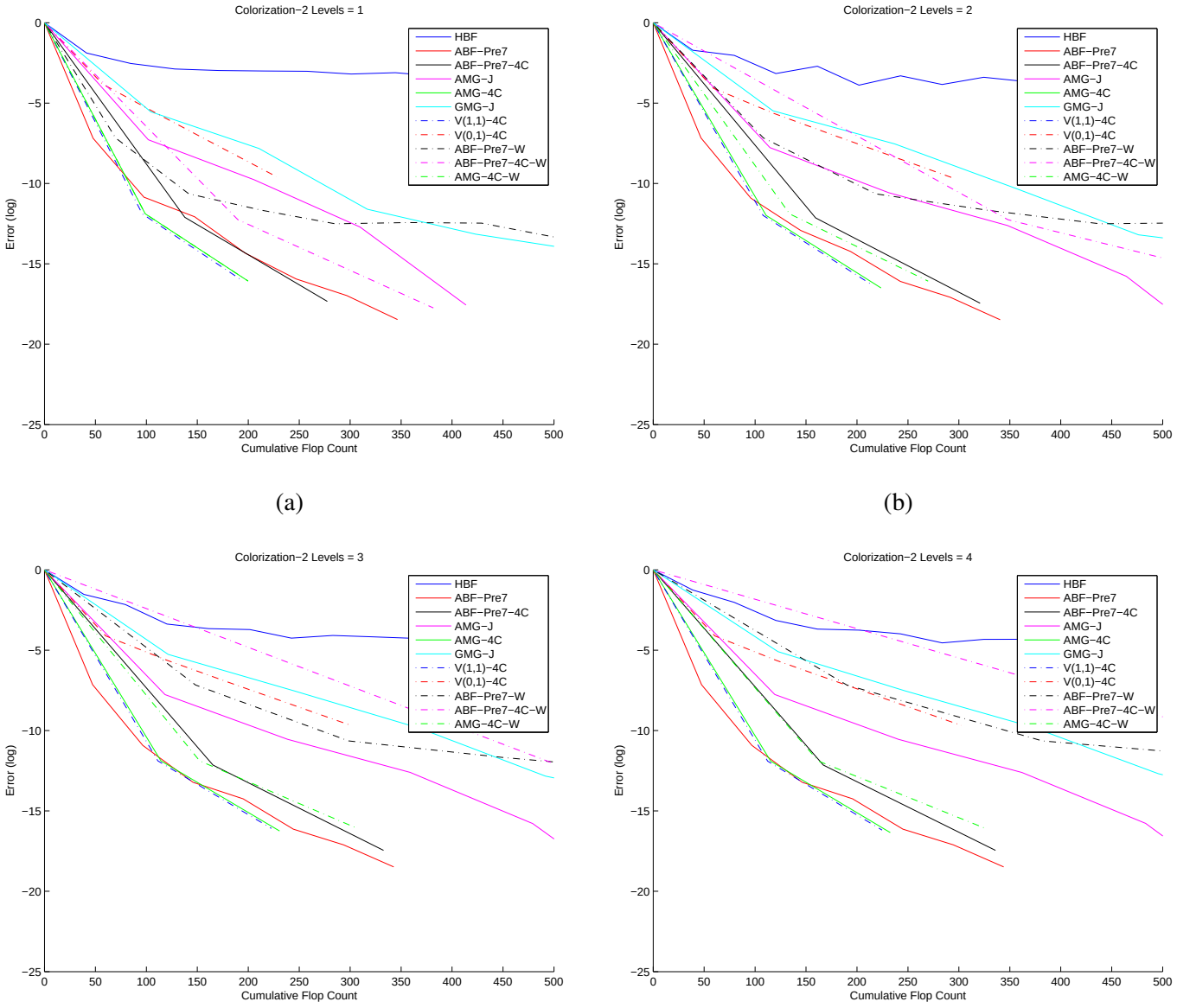


Figure 159: Log error vs. flops, Colorization 9-pt, 32×32 image

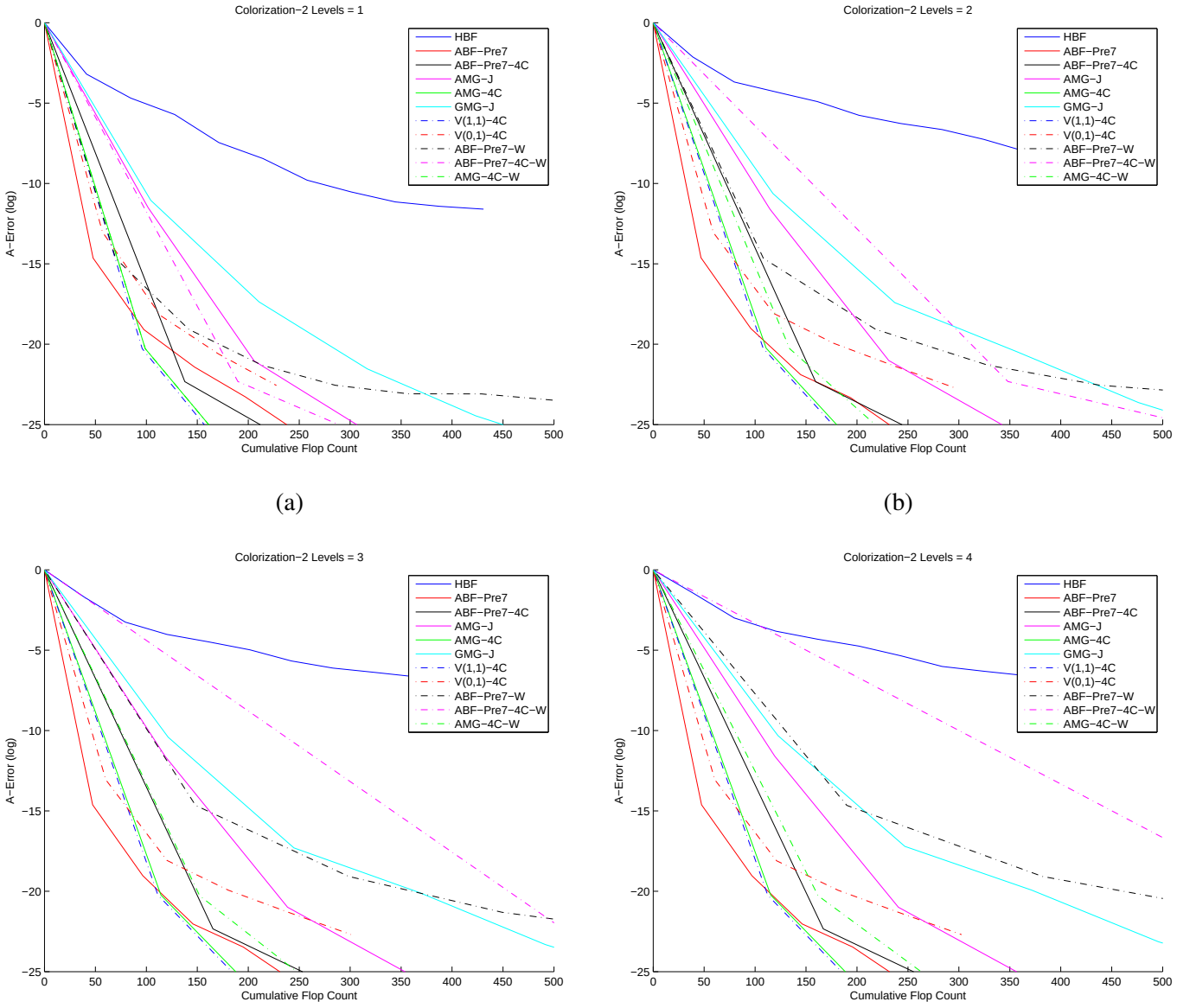


Figure 160: Log A-error vs. flops, Colorization 9-pt, 32×32 image

	$L=4$				$L=5$				$L=6$				$L=7$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.83	0.19	40.8	0.47	0.71	0.34	40.7	0.82	0.66	0.41	40.7	1.01	0.65	0.43	40.7	1.06
ABF-Pre7	0.13	2.07	49.3	4.19	0.13	2.06	49.3	4.18	0.13	2.06	49.3	4.17	0.13	2.06	49.3	4.17
ABF-Pre7-4C	0.01	4.81	168.6	2.85	0.01	4.81	169.0	2.85	0.01	4.81	169.1	2.85	0.01	4.81	169.1	2.85
AMG-J	0.10	2.26	121.3	1.87	0.11	2.23	121.5	1.84	0.11	2.22	121.5	1.83	0.11	2.22	121.6	1.83
AMG-4C	0.04	3.24	117.1	2.77	0.04	3.23	117.3	2.75	0.04	3.22	117.4	2.75	0.04	3.22	117.4	2.75
GMG-J	0.21	1.54	124.5	1.23	0.21	1.54	124.8	1.23	0.22	1.52	124.8	1.22	0.22	1.51	124.9	1.21
V(1,1)-4C	0.07	2.60	112.6	2.31	0.08	2.51	112.8	2.23	0.08	2.51	112.9	2.22	0.08	2.51	112.9	2.22
V(0,1)-4C	0.16	1.81	60.7	2.98	0.18	1.73	60.8	2.85	0.19	1.67	60.8	2.75	0.19	1.68	60.8	2.76
CMG	0.10	2.33	112.3	2.07	0.10	2.33	118.0	1.97	0.10	2.33	123.4	1.89	0.10	2.33	128.5	1.81
ABF-Pre7-W	0.10	2.27	205.3	1.11	0.10	2.27	245.8	0.92	0.10	2.27	291.4	0.78	0.10	2.27	348.8	0.65
ABF-Pre7-4C-W	0.00	5.37	689.9	0.78	0.00	5.37	853.6	0.63	0.00	5.37	1029.6	0.52	0.00	5.37	1234.8	0.44
AMG-4C-W	0.01	4.64	164.6	2.82	0.01	4.64	169.1	2.75	0.01	4.64	171.6	2.71	0.01	4.64	173.1	2.68

(a) empirical convergence results

	$L=4$						$L=5$						$L=6$						$L=7$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	270.64	27913.78	0.89	0.12	40.8	0.30	263.66	7767.93	0.88	0.12	40.7	0.30	201.41	89.84	0.87	0.14	40.7	0.35	187.89	20.31	0.86	0.15	40.7	0.36
ABF-Pre7	6.01	481.80	0.42	0.87	49.3	1.76	6.03	389.49	0.42	0.86	49.3	1.75	6.03	375.97	0.42	0.86	49.3	1.75	6.03	375.94	0.42	0.86	49.3	1.75
ABF-Pre7-4C	1.73	481.80	0.14	1.99	168.6	1.18	1.73	389.49	0.14	1.99	169.0	1.18	1.73	375.97	0.14	1.99	169.1	1.18	1.73	375.94	0.14	1.99	169.1	1.18
AMG-J	1.62	183349.89	0.12	2.12	121.3	1.75	1.76	220.88	0.14	1.96	121.5	1.61	1.81	181.83	0.15	1.91	121.5	1.58	1.81	181.83	0.15	1.91	121.6	1.57
AMG-4C	1.40	183349.89	0.08	2.48	117.1	2.12	1.47	220.88	0.10	2.35	117.3	2.00	1.47	181.83	0.10	2.34	117.4	2.00	1.47	181.83	0.10	2.34	117.4	2.00
GMG-J	4.72	27913.78	0.37	1.00	124.5	0.80	4.87	7767.93	0.38	0.98	124.8	0.78	4.81	89.84	0.37	0.98	124.8	0.79	4.81	20.31	0.37	0.98	124.9	0.79
V(1,1)-4C	1.40	183349.89	0.08	2.48	112.6	2.21	1.47	220.88	0.10	2.35	112.8	2.08	1.47	181.83	0.10	2.34	112.9	2.08	1.47	181.83	0.10	2.34	112.9	2.08
V(0,1)-4C	1.56	183349.89	0.11	2.21	60.7	3.63	1.66	220.88	0.13	2.07	60.8	3.41	1.83	181.83	0.15	1.90	60.8	3.12	1.83	181.83	0.15	1.90	60.8	3.12
CMG	2.03	77.97	0.17	1.74	112.3	1.55	2.03	15.16	0.17	1.74	118.0	1.48	2.03	6.44	0.17	1.74	123.4	1.41	2.03	2.31	0.17	1.74	128.5	1.36
ABF-Pre7-W	-	-	-	-	205.3	-	-	-	-	-	245.8	-	-	-	-	-	291.4	-	-	-	-	-	348.8	-
ABF-Pre7-4C-W	-	-	-	-	689.9	-	-	-	-	-	853.6	-	-	-	-	-	1029.6	-	-	-	-	-	1234.8	-
AMG-4C-W	-	-	-	-	164.6	-	-	-	-	-	169.1	-	-	-	-	-	171.6	-	-	-	-	-	173.1	-

Original condition number =5124.5

(b) theoretical convergence results

Table 121: Colorization 9-pt, 256×256 image

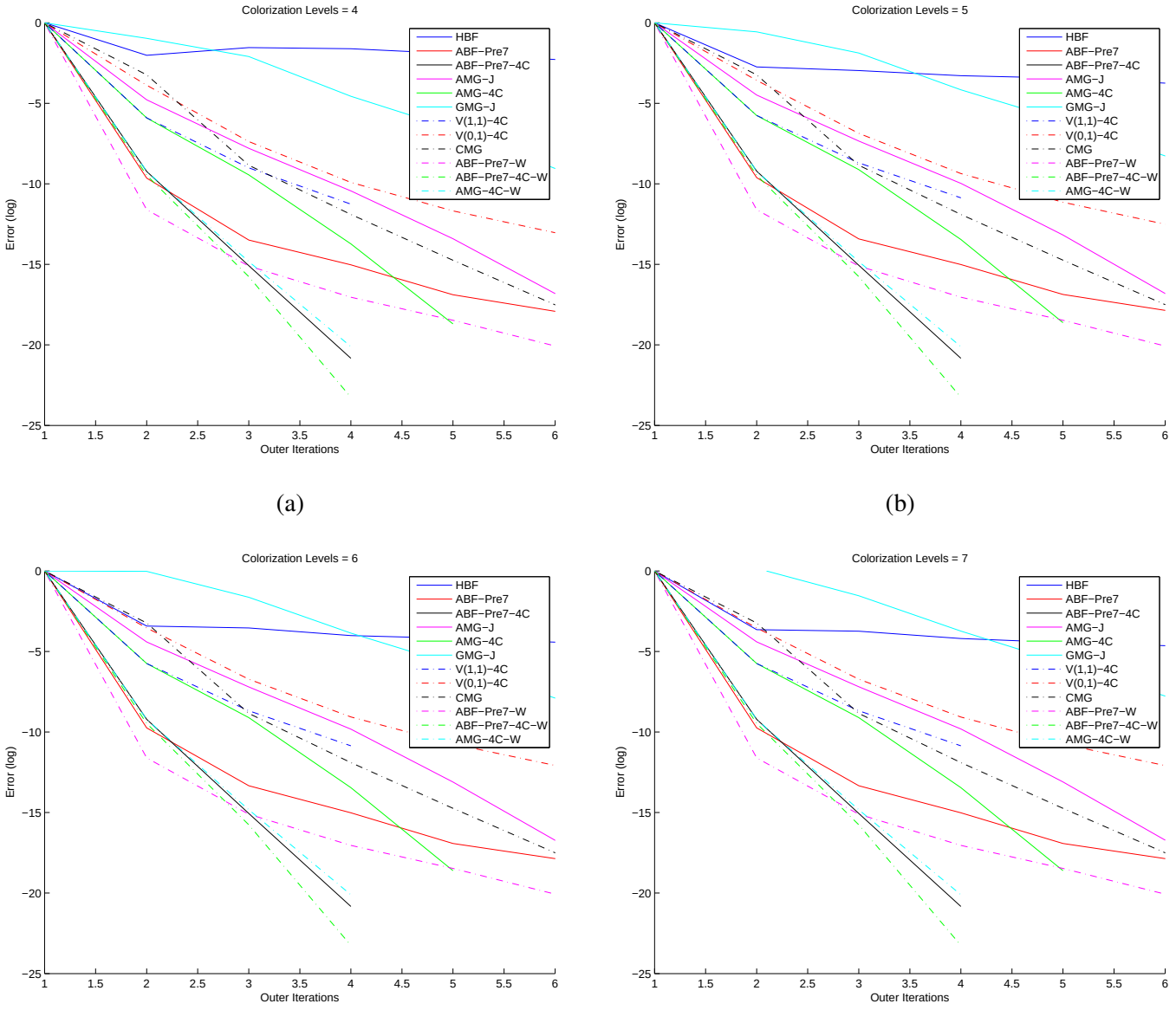


Figure 161: Log error vs. its, Colorization 9-pt, 256×256 image

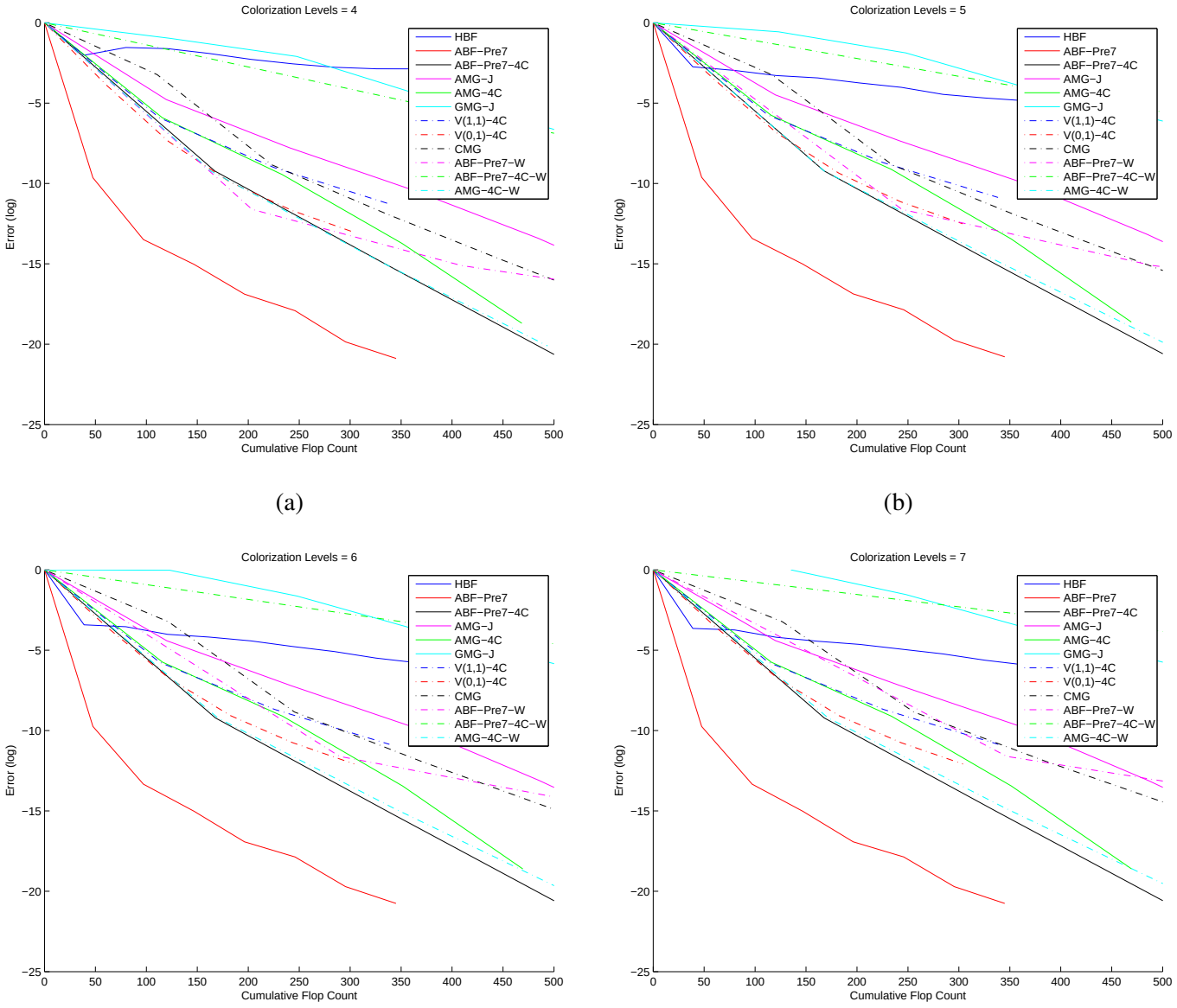


Figure 162: Log error vs. flops, Colorization 9-pt, 256×256 image

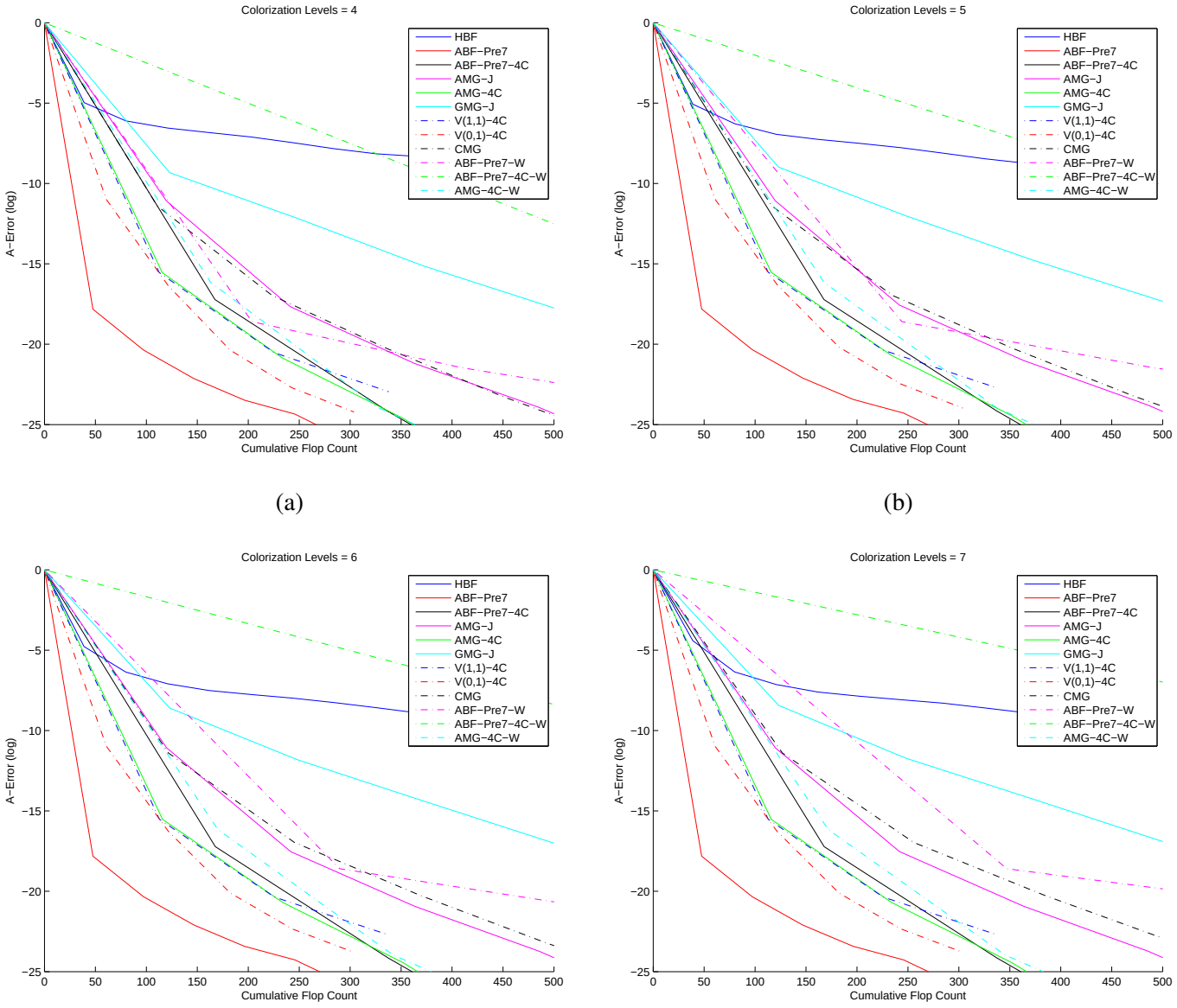


Figure 163: Log A-error vs. flops, Colorization 9-pt, 256×256 image

	$L=3$				$L=4$				$L=5$				$L=6$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.97	0.03	41.1	0.08	0.85	0.17	40.8	0.41	0.81	0.21	40.7	0.51	0.71	0.34	40.7	0.82
ABF-Pre7	0.16	1.85	49.6	3.72	0.16	1.83	49.3	3.71	0.16	1.82	49.3	3.68	0.16	1.82	49.3	3.68
ABF-Pre7-4C	0.02	3.71	167.5	2.21	0.03	3.68	168.7	2.18	0.03	3.68	169.0	2.17	0.03	3.68	169.1	2.18
AMG-J	0.14	2.00	120.7	1.66	0.21	1.55	121.3	1.28	0.27	1.32	121.5	1.09	0.27	1.30	121.6	1.07
AMG-4C	0.10	2.35	116.6	2.01	0.18	1.70	117.2	1.45	0.24	1.45	117.4	1.23	0.25	1.38	117.5	1.17
GMG-J	0.29	1.23	124.0	0.99	0.34	1.08	124.6	0.86	0.35	1.05	124.8	0.84	0.36	1.04	124.9	0.83
V(1,1)-4C	0.21	1.55	112.1	1.38	0.27	1.31	112.6	1.17	0.31	1.18	112.8	1.05	0.32	1.13	112.9	1.00
V(0,1)-4C	0.21	1.56	60.8	2.57	0.29	1.23	60.7	2.02	0.36	1.03	60.8	1.69	0.39	0.94	60.8	1.55
CMG	0.09	2.41	106.4	2.27	0.09	2.37	112.8	2.10	0.09	2.36	118.9	1.99	0.09	2.36	125.2	1.89
ABF-Pre7-W	0.29	1.22	167.1	0.73	0.29	1.22	203.7	0.60	0.29	1.22	243.5	0.50	0.29	1.22	286.5	0.43
ABF-Pre7-4C-W	0.02	4.02	528.4	0.76	0.02	4.02	685.9	0.59	0.02	4.02	846.6	0.47	0.02	4.02	1018.5	0.39
AMG-4C-W	0.05	2.96	156.3	1.89	0.05	2.90	164.5	1.77	0.06	2.83	169.0	1.67	0.05	2.92	171.4	1.70

(a) empirical convergence results

	$L=3$						$L=4$						$L=5$						$L=6$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	2105.58	439932.04	0.96	0.04	41.1	0.11	271.83	345545.75	0.89	0.12	40.8	0.30	235.35	144112.25	0.88	0.13	40.7	0.32	187.60	11036.40	0.86	0.15	40.7	0.36
ABF-Pre7	6.81	15905.58	0.45	0.81	49.6	1.63	7.25	8974.69	0.46	0.78	49.3	1.58	7.73	7265.83	0.47	0.75	49.3	1.53	8.19	4725.00	0.48	0.73	49.3	1.48
ABF-Pre7-4C	1.78	15905.58	0.14	1.94	167.5	1.16	1.78	8974.69	0.14	1.94	168.7	1.15	1.78	7265.83	0.14	1.95	169.0	1.15	1.78	4725.00	0.14	1.95	169.1	1.15
AMG-J	3.48	23914988.89	0.30	1.20	120.7	0.99	4.91	327176.46	0.38	0.97	121.3	0.80	5.81	78686.61	0.41	0.88	121.5	0.73	5.89	64637.27	0.42	0.88	121.6	0.72
AMG-4C	3.32	23914988.89	0.29	1.23	116.6	1.06	4.86	327176.46	0.38	0.98	117.2	0.83	5.92	78686.61	0.42	0.87	117.4	0.74	6.29	64637.27	0.43	0.84	117.5	0.72
GMG-J	5.57	439932.04	0.40	0.90	124.0	0.73	6.24	345545.75	0.43	0.85	124.6	0.68	6.77	144112.25	0.44	0.81	124.8	0.65	6.97	11036.40	0.45	0.80	124.9	0.64
V(1,1)-4C	3.32	23914988.89	0.29	1.23	112.1	1.10	4.86	327176.46	0.38	0.98	112.6	0.87	5.92	78686.61	0.42	0.87	112.8	0.77	6.29	64637.27	0.43	0.84	112.9	0.75
V(0,1)-4C	3.97	23914988.89	0.33	1.10	60.8	1.82	6.28	327176.46	0.43	0.84	60.7	1.39	6.33	78686.61	0.43	0.84	60.8	1.38	5.53	64637.27	0.40	0.91	60.8	1.49
CMG	2.07	1702.42	0.18	1.71	106.4	1.61	2.09	454.76	0.18	1.70	112.8	1.51	2.10	120.31	0.18	1.70	118.9	1.43	2.10	31.06	0.18	1.70	125.2	1.36
ABF-Pre7-W	-	-	-	-	167.1	-	-	-	-	-	203.7	-	-	-	-	-	243.5	-	-	-	-	-	286.5	-
ABF-Pre7-4C-W	-	-	-	-	528.4	-	-	-	-	-	685.9	-	-	-	-	-	846.6	-	-	-	-	-	1018.5	-
AMG-4C-W	-	-	-	-	156.3	-	-	-	-	-	164.5	-	-	-	-	-	169.0	-	-	-	-	-	171.4	-

Original condition number =10400.3

(b) theoretical convergence results

Table 122: Colorization 9-pt, 239×319 image

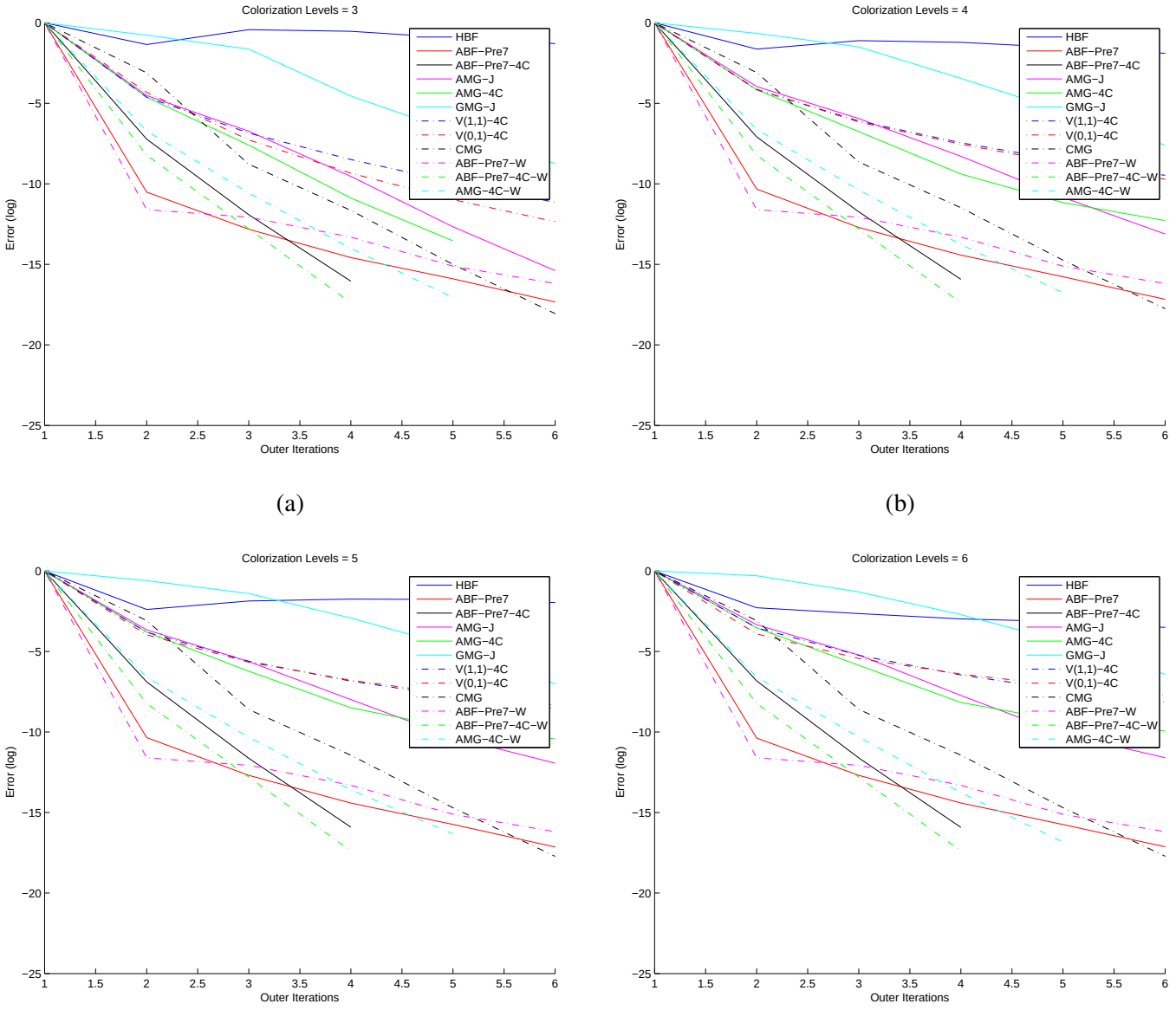


Figure 164: Log error vs. its, Colorization 9-pt, 239×319 image

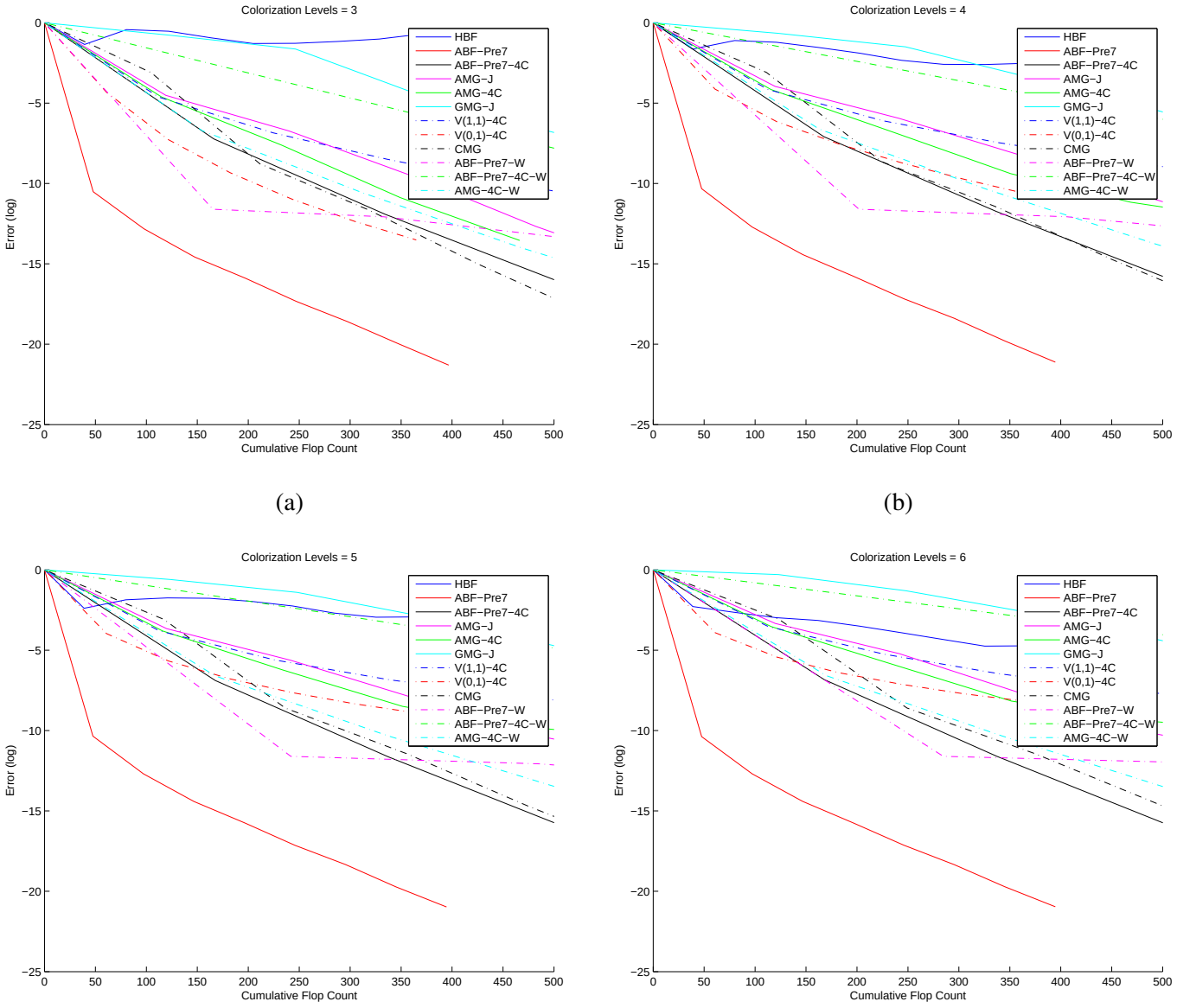
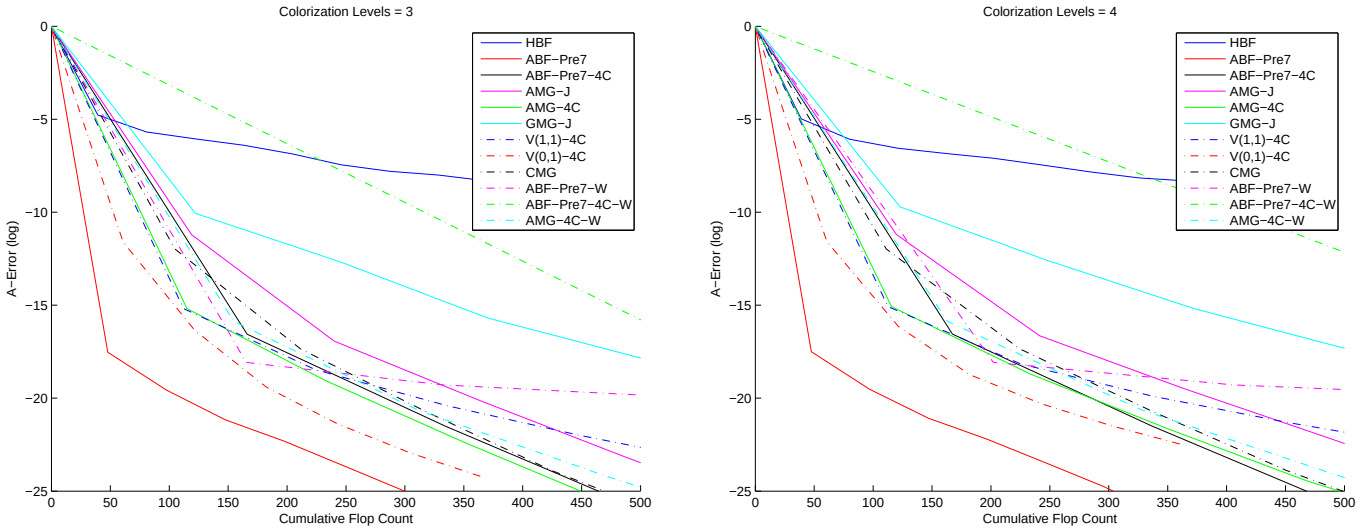
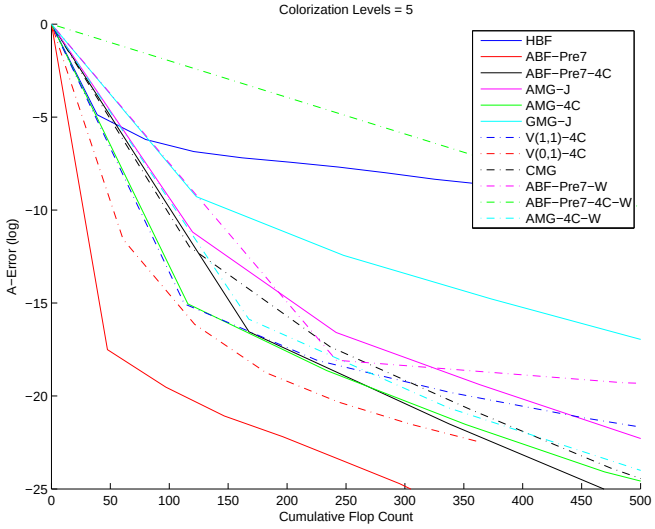


Figure 165: Log error vs. flops, Colorization 9-pt, 239×319 image

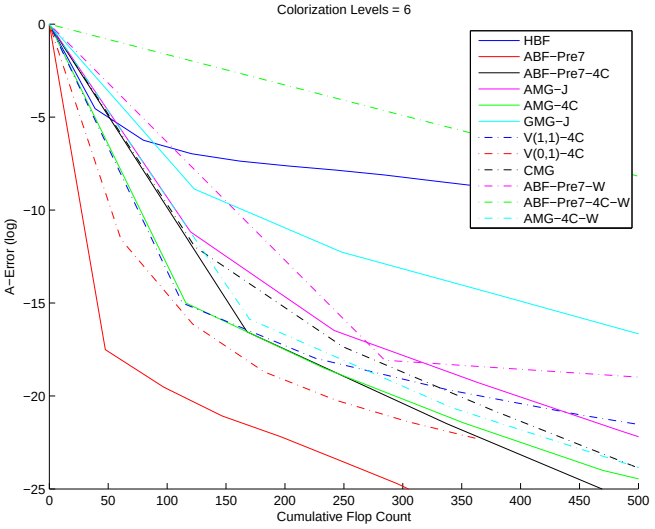


(a)

(b)



(c)



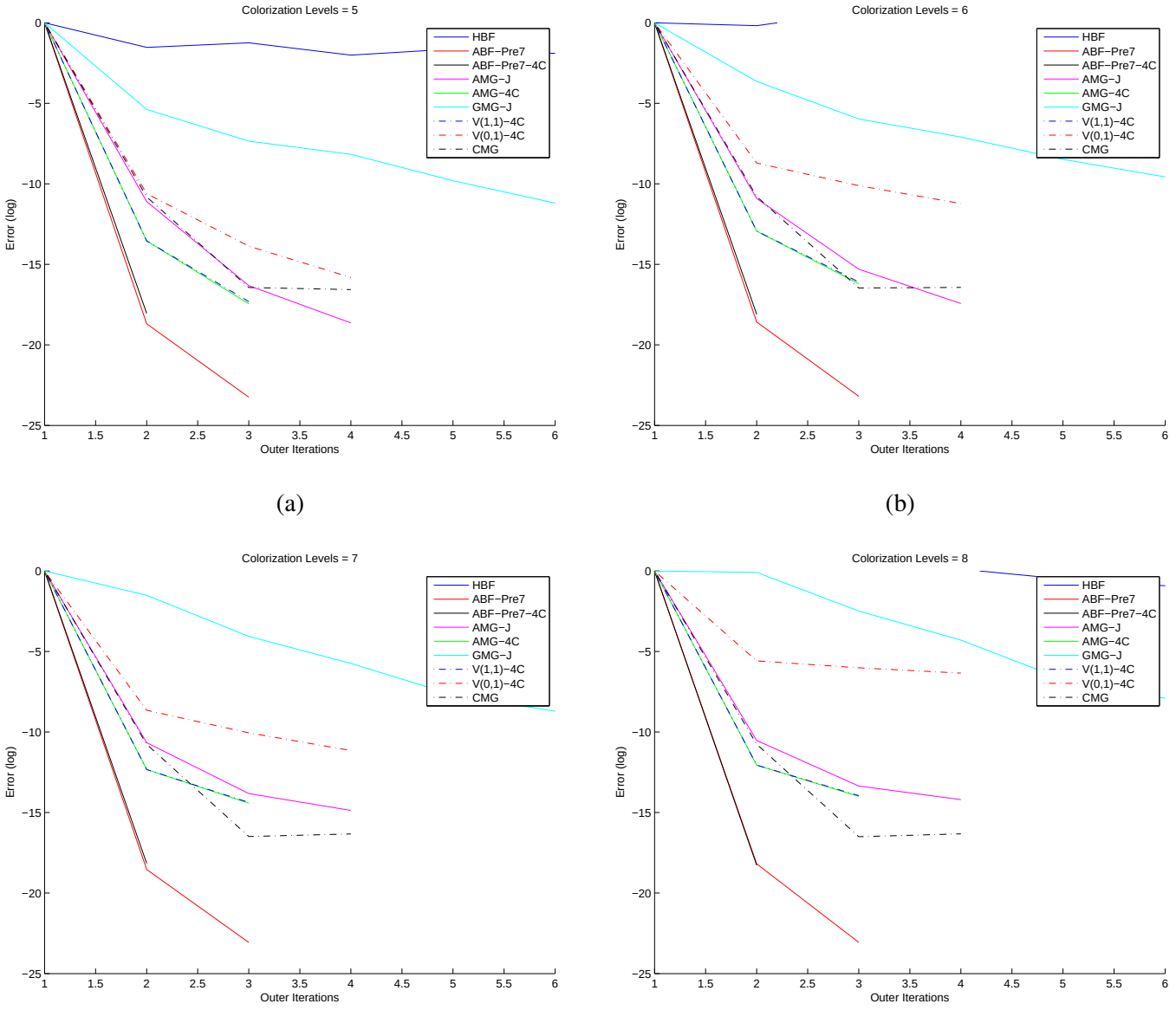
(d)

Figure 166: Log A-error vs. flops, Colorization 9-pt, 239×319 image

	$L=5$				$L=6$				$L=7$				$L=8$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.84	0.17	40.7	0.42	0.93	0.07	40.6	0.17	0.84	0.17	40.6	0.43	0.80	0.23	40.6	0.56
ABF-Pre7	0.00	8.06	48.5	16.60	0.00	8.04	48.5	16.57	0.00	8.00	48.5	16.48	0.00	7.99	48.5	16.48
ABF-Pre7-4C	0.00	12.51	167.4	7.47	0.00	12.55	167.5	7.49	0.00	12.58	167.5	7.51	0.00	12.67	167.5	7.56
AMG-J	0.01	4.31	121.0	3.56	0.02	4.03	121.0	3.33	0.03	3.44	121.0	2.84	0.04	3.28	121.0	2.71
AMG-4C	0.00	6.05	116.6	5.18	0.00	5.62	116.7	4.81	0.01	5.00	116.7	4.28	0.01	4.85	116.7	4.16
GMG-J	0.19	1.66	124.5	1.33	0.24	1.44	124.5	1.16	0.27	1.33	124.6	1.06	0.29	1.23	124.6	0.98
V(1,1)-4C	0.00	6.00	112.6	5.33	0.00	5.58	112.7	4.95	0.01	4.98	112.7	4.42	0.01	4.83	112.7	4.29
V(0,1)-4C	0.03	3.65	60.7	6.02	0.07	2.59	60.7	4.28	0.08	2.58	60.7	4.25	0.23	1.47	60.7	2.42
CMG	0.02	3.83	104.8	3.65	0.02	3.80	105.0	3.61	0.02	3.77	105.7	3.57	0.02	3.77	106.3	3.55

(a) empirical convergence results

Table 123: Colorization 9-pt, 1854×2048 image



(a) (b)

(c) (d)

Figure 167: Log error vs. its, Colorization 9-pt, 1854×2048 image

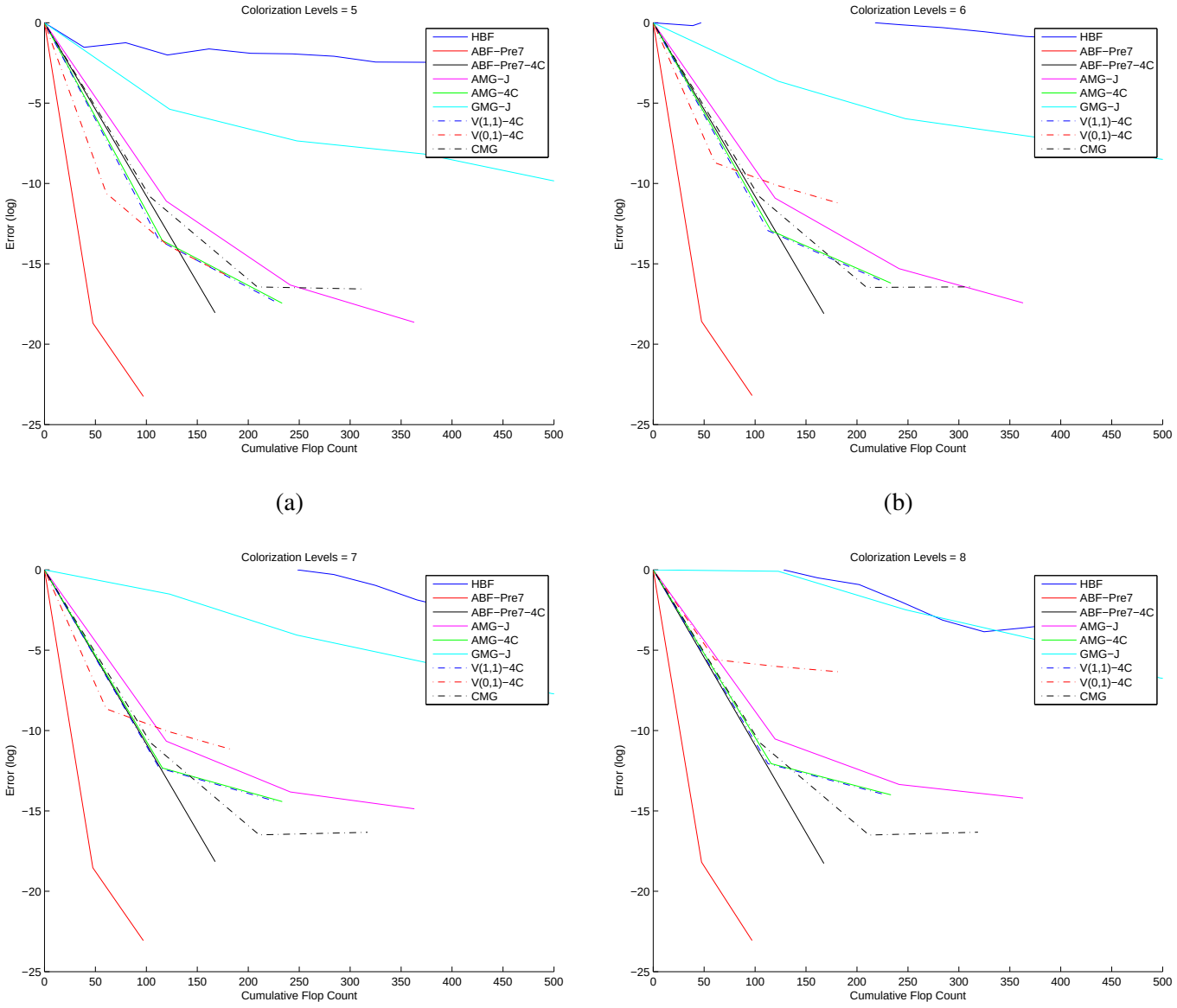


Figure 168: Log error vs. flops, Colorization 9-pt, 1854×2048 image

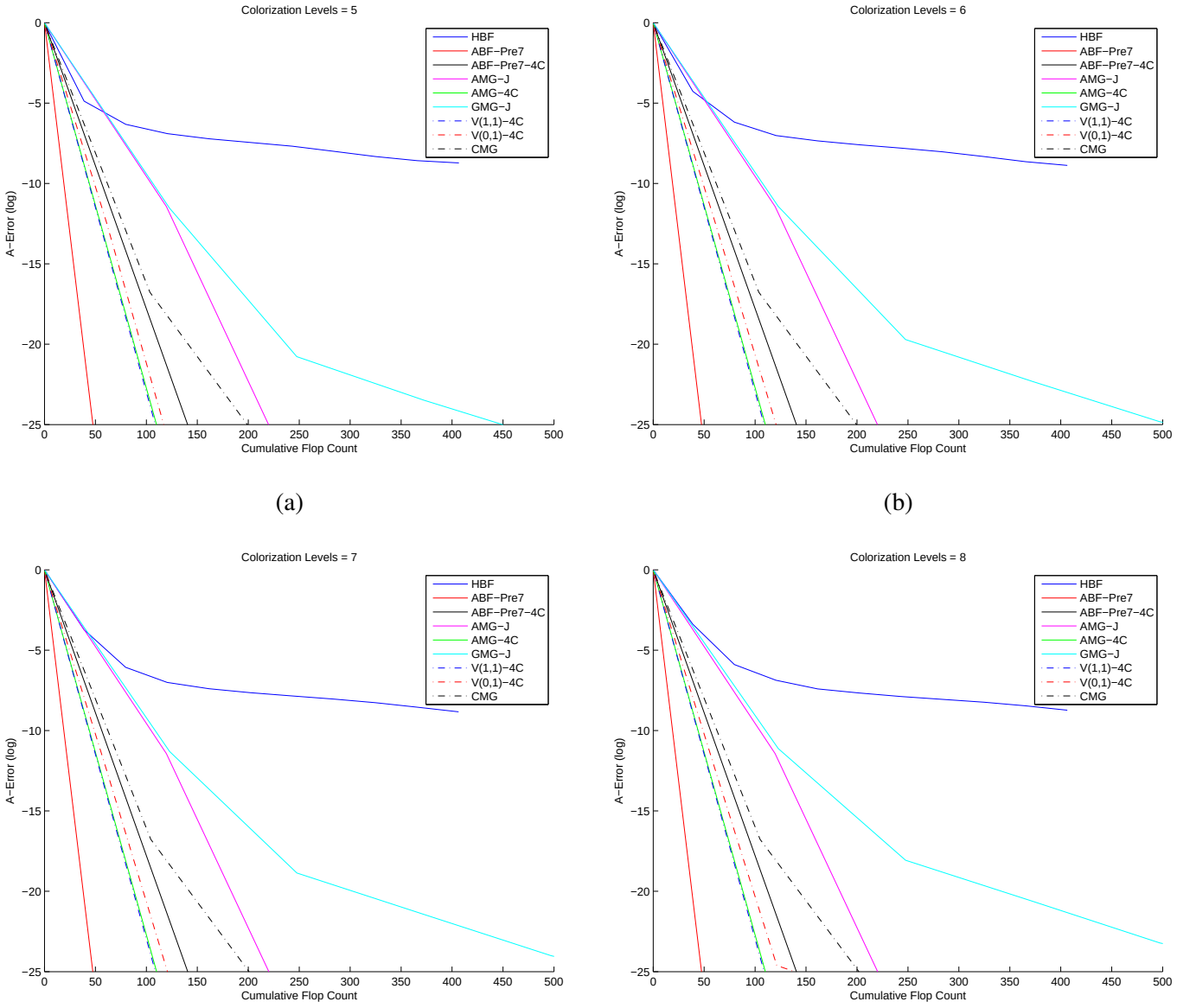


Figure 169: Log A-error vs. flops, Colorization 9-pt, 1854×2048 image

Algorithm	$L=1$				$L=2$				$L=3$				$L=4$			
	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.38	0.96	43.7	2.20	0.47	0.76	35.6	2.14	0.49	0.71	34.6	2.04	0.51	0.68	34.6	1.97
ABF-Pre7	0.18	1.70	50.0	3.40	0.19	1.64	43.5	3.77	0.19	1.65	42.9	3.83	0.19	1.64	43.0	3.82
ABF-Pre7-4C	0.01	4.39	80.0	5.49	0.01	4.21	97.4	4.32	0.01	4.28	103.5	4.13	0.01	4.26	105.7	4.03
AMG-J	0.12	2.16	81.0	2.66	0.13	2.01	89.2	2.26	0.18	1.69	92.7	1.82	0.25	1.38	94.1	1.47
AMG-4C	0.08	2.53	77.0	3.28	0.10	2.30	84.1	2.74	0.13	2.06	87.4	2.36	0.18	1.69	88.6	1.91
GMG-J	0.24	1.43	83.7	1.70	0.36	1.02	92.6	1.10	0.37	0.98	96.4	1.02	0.38	0.97	97.8	0.99
V(1,1)-4C	0.21	1.57	72.2	2.18	0.22	1.50	79.3	1.89	0.30	1.21	82.6	1.46	0.46	0.78	83.8	0.93
V(0,1)-4C	0.25	1.38	49.2	2.80	0.27	1.31	45.9	2.86	0.32	1.14	46.3	2.45	0.50	0.69	46.7	1.48
CMG	0.06	2.88	43.2	6.67	0.09	2.41	44.7	5.40	0.09	2.37	46.6	5.08	0.09	2.36	48.5	4.87
ABF-Pre7-W	0.25	1.37	80.2	1.71	0.25	1.38	120.2	1.15	0.25	1.38	171.0	0.81	0.25	1.38	240.3	0.57
ABF-Pre7-4C-W	0.00	8.61	139.9	6.16	0.00	8.59	307.5	2.79	0.00	8.59	502.1	1.71	0.00	8.59	751.2	1.14
AMG-4C-W	0.08	2.53	77.0	3.28	0.09	2.43	110.4	2.20	0.09	2.41	131.4	1.83	0.10	2.28	145.5	1.57

(a) empirical convergence results

Algorithm	$L=1$						$L=2$						$L=3$						$L=4$					
	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	13.95	3735.98	0.58	0.55	43.7	1.26	22.72	554.55	0.65	0.43	35.6	1.20	37.24	118.28	0.72	0.33	34.6	0.95	39.55	20.30	0.73	0.32	34.6	0.93
ABF-Pre7	6.61	2515.63	0.44	0.82	50.0	1.64	6.70	157.83	0.44	0.81	43.5	1.87	8.75	16.94	0.49	0.70	42.9	1.64	8.72	5.76	0.49	0.70	43.0	1.64
ABF-Pre7-4C	1.39	2515.63	0.08	2.50	80.0	3.13	1.37	157.83	0.08	2.54	97.4	2.61	1.36	16.94	0.08	2.57	103.5	2.48	1.36	5.76	0.08	2.58	105.7	2.44
AMG-J	2.22	3451.96	0.20	1.63	81.0	2.01	2.14	2089031.46	0.19	1.67	89.2	1.87	4.05	3604.94	0.34	1.09	92.7	1.18	5.74	22568.81	0.41	0.89	94.1	0.94
AMG-4C	1.80	3451.96	0.15	1.92	77.0	2.49	1.80	2089031.46	0.15	1.92	84.1	2.28	3.56	3604.94	0.31	1.18	87.4	1.35	4.54	22568.81	0.36	1.02	88.6	1.15
GMG-J	4.77	3735.98	0.37	0.99	83.7	1.18	9.21	554.55	0.50	0.68	92.6	0.74	13.65	118.28	0.57	0.56	96.4	0.58	13.19	20.30	0.57	0.57	97.8	0.58
V(1,1)-4C	1.80	3451.96	0.15	1.92	72.2	2.66	1.80	2089031.46	0.15	1.92	79.3	2.42	3.56	3604.94	0.31	1.18	82.6	1.43	4.54	22568.81	0.36	1.02	83.8	1.21
V(0,1)-4C	1.91	3451.96	0.16	1.83	49.2	3.72	2.27	2089031.46	0.20	1.60	45.9	3.48	4.50	3604.94	0.36	1.02	46.3	2.21	5.99	22568.81	0.42	0.87	46.7	1.86
CMG	1.78	6596.32	0.14	1.94	43.2	4.50	2.71	602.25	0.24	1.41	44.7	3.15	3.02	122.51	0.27	1.31	46.6	2.81	3.07	29.15	0.27	1.30	48.5	2.67
ABF-Pre7-W	-	2515.63	-	-	80.2	-	-	157.83	-	-	120.2	-	-	16.94	-	-	171.0	-	-	5.76	-	-	240.3	-
ABF-Pre7-4C-W	1.02	2515.63	0.00	5.52	139.9	3.95	1.02	157.83	0.00	5.54	307.5	1.80	1.02	16.94	0.00	5.54	502.1	1.10	1.02	5.76	0.00	5.54	751.2	0.74
AMG-4C-W	1.80	3451.96	0.15	1.92	77.0	2.49	1.78	2089031.46	0.14	1.95	110.4	1.76	2.03	3604.94	0.17	1.74	131.4	1.33	2.30	22568.81	0.21	1.58	145.5	1.09

Original condition number =15716.9

(b) theoretical convergence results

Table 124: EPD, 33×33 image

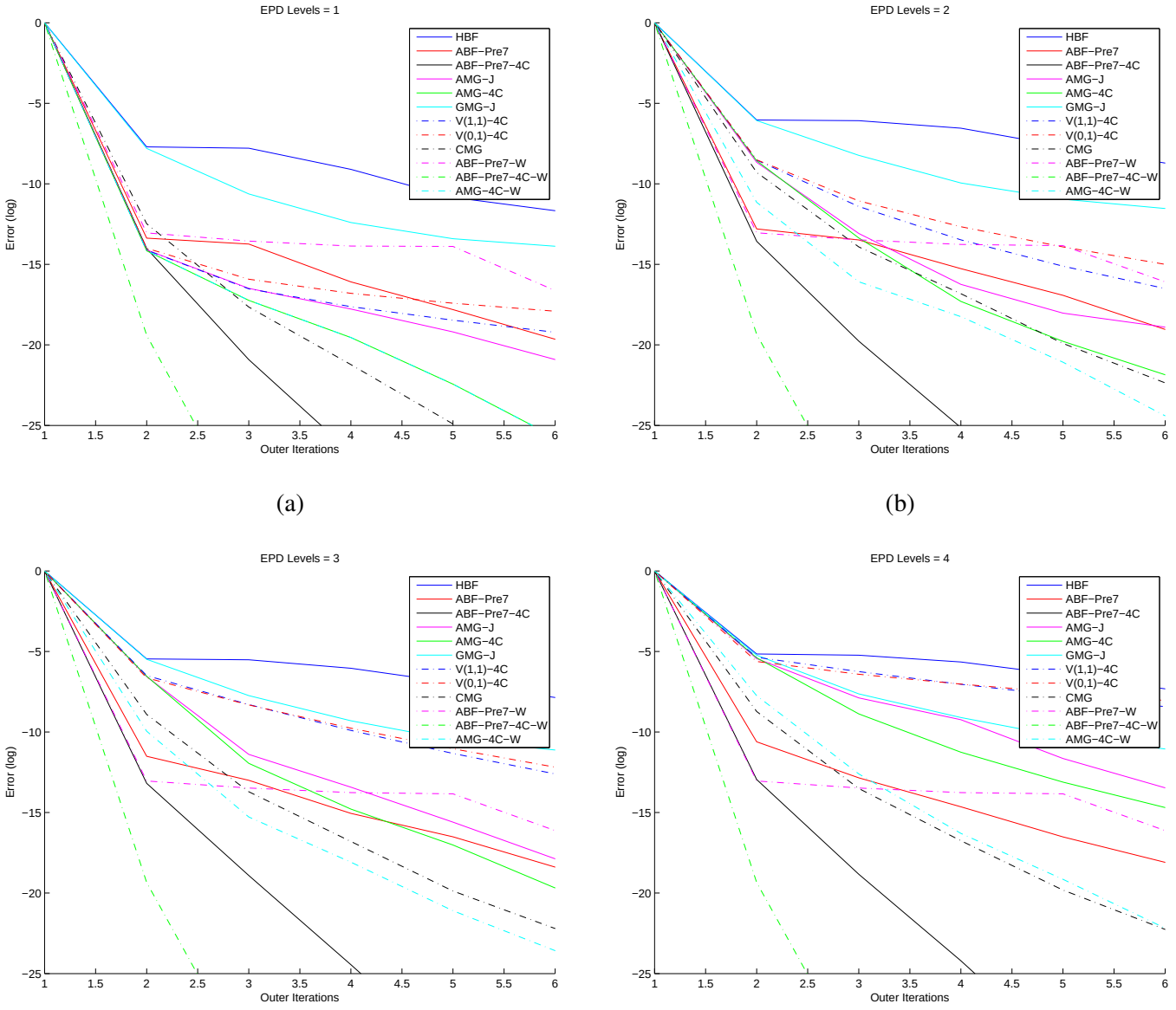


Figure 170: Log error vs. its, EPD, 33×33 image

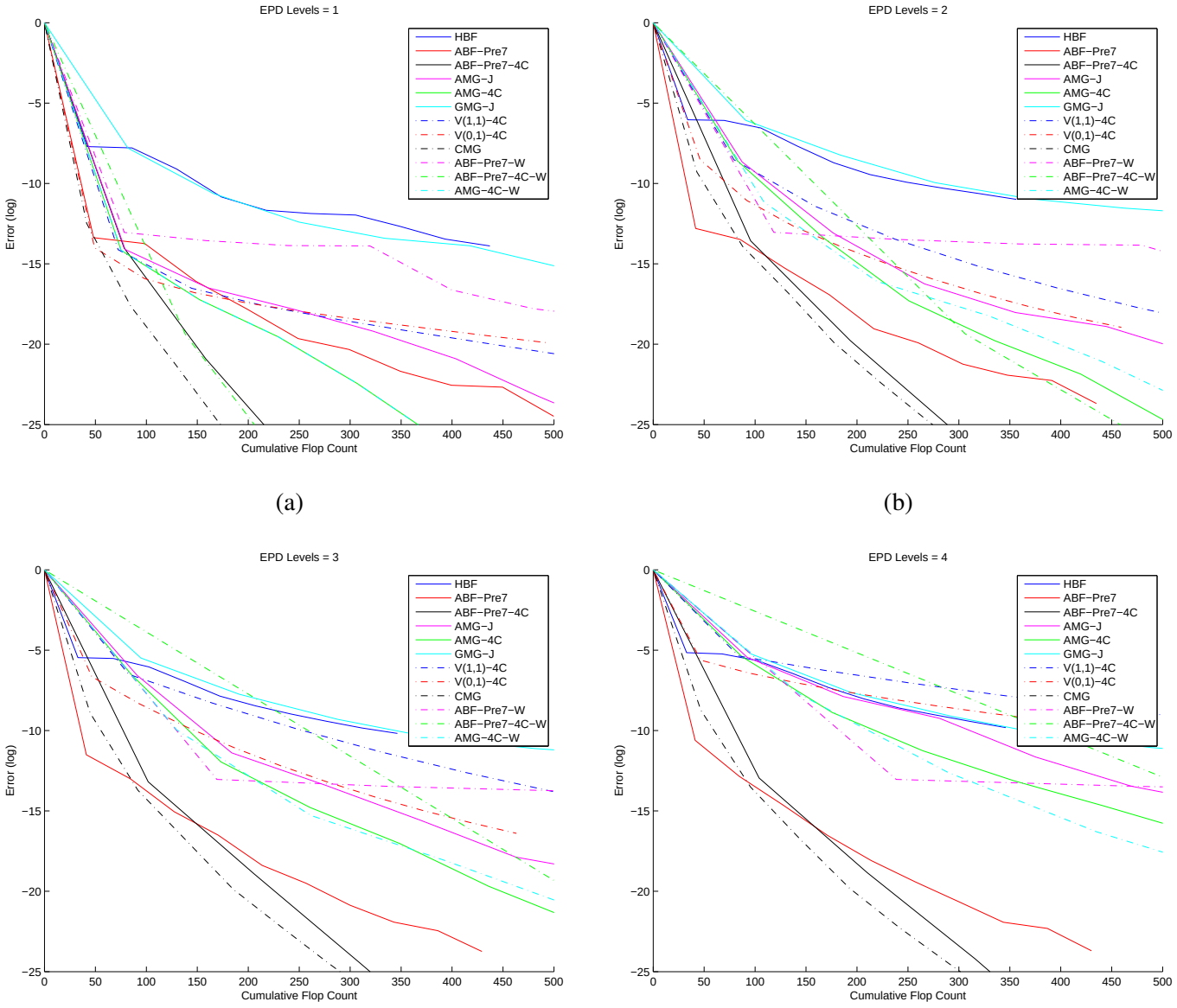


Figure 171: Log error vs. flops, EPD, 33×33 image

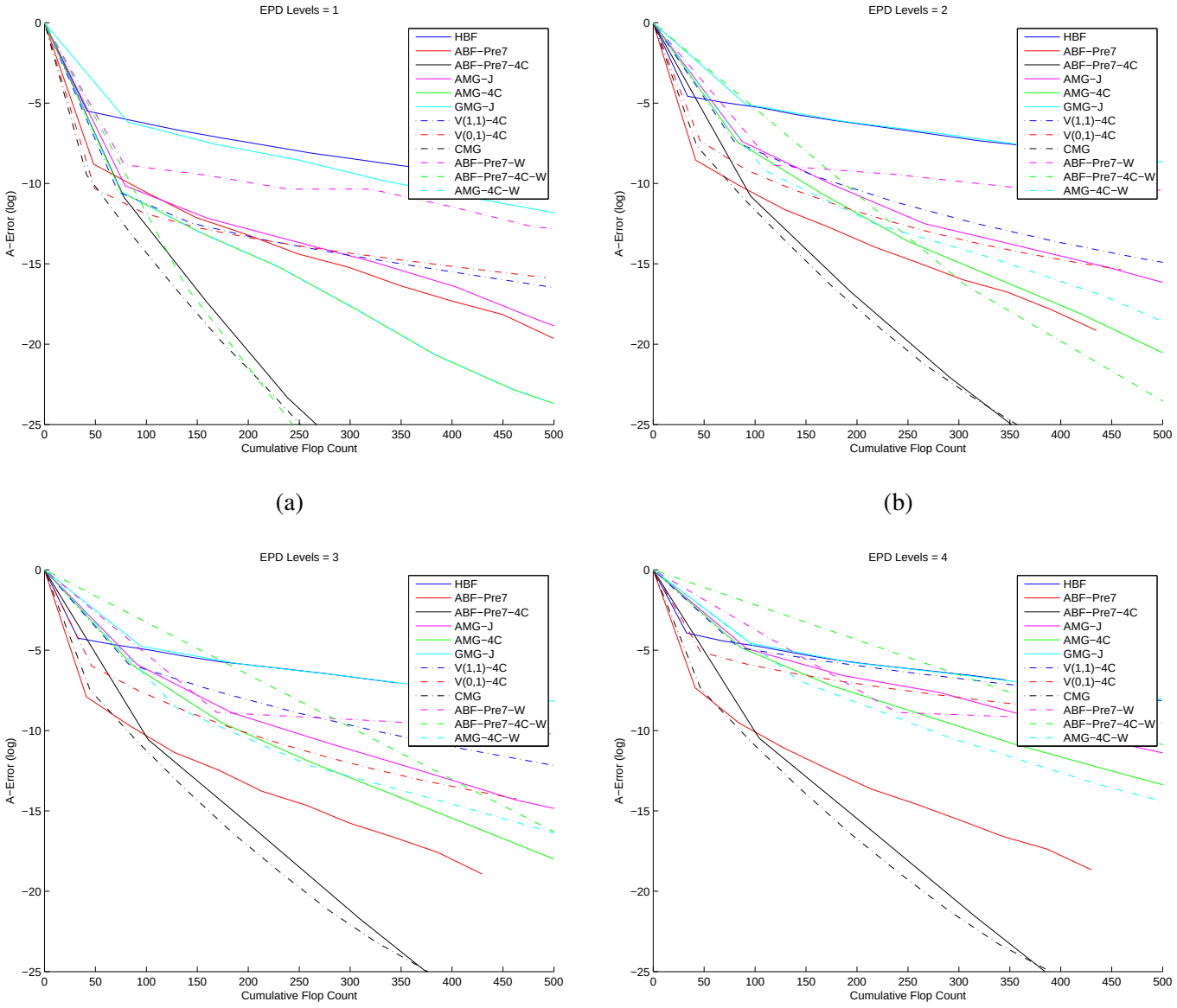


Figure 172: Log A-error vs. flops, EPD, 33×33 image

Algorithm	$L=1$				$L=2$				$L=3$				$L=4$			
	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.47	0.75	43.7	1.73	0.59	0.53	35.6	1.48	0.61	0.50	34.6	1.44	0.61	0.50	34.6	1.43
ABF-Pre7	0.20	1.61	50.0	3.23	0.22	1.53	43.5	3.51	0.22	1.52	42.9	3.54	0.22	1.50	43.0	3.49
ABF-Pre7-4C	0.02	4.10	80.0	5.13	0.02	3.87	97.4	3.97	0.02	3.80	103.5	3.67	0.02	3.79	105.7	3.59
AMG-J	0.15	1.89	81.0	2.33	0.20	1.59	89.2	1.79	0.21	1.58	92.7	1.70	0.22	1.52	94.1	1.62
AMG-4C	0.08	2.48	77.0	3.22	0.12	2.12	84.1	2.52	0.13	2.03	87.4	2.32	0.13	2.04	88.6	2.31
GMG-J	0.28	1.26	83.7	1.50	0.44	0.81	92.6	0.88	0.46	0.77	96.4	0.80	0.46	0.77	97.8	0.79
V(1,1)-4C	0.21	1.55	72.2	2.15	0.32	1.15	79.3	1.45	0.36	1.03	82.6	1.24	0.41	0.89	83.8	1.06
V(0,1)-4C	0.25	1.39	49.2	2.82	0.39	0.95	45.9	2.07	0.45	0.81	46.3	1.75	0.48	0.73	46.7	1.55
CMG	0.09	2.38	42.9	5.54	0.13	2.03	43.8	4.63	0.13	2.02	46.2	4.37	0.13	2.01	48.6	4.14
ABF-Pre7-W	0.23	1.49	80.2	1.86	0.24	1.42	120.2	1.18	0.24	1.42	171.0	0.83	0.24	1.42	240.3	0.59
ABF-Pre7-4C-W	0.00	7.47	139.9	5.34	0.00	7.36	307.5	2.39	0.00	7.36	502.1	1.47	0.00	7.36	751.2	0.98
AMG-4C-W	0.08	2.48	77.0	3.22	0.10	2.29	110.4	2.07	0.10	2.30	131.4	1.75	0.10	2.30	145.5	1.58

(a) empirical convergence results

Algorithm	$L=1$						$L=2$						$L=3$						$L=4$					
	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	12.48	5220.07	0.56	0.58	43.7	1.33	23.63	619.11	0.66	0.42	35.6	1.17	29.14	95.68	0.69	0.37	34.6	1.08	33.98	15.33	0.71	0.35	34.6	1.00
ABF-Pre7	4.06	4589.49	0.34	1.09	50.0	2.18	5.30	197.41	0.39	0.93	43.5	2.14	5.98	16.87	0.42	0.87	42.9	2.02	6.23	3.62	0.43	0.85	43.0	1.97
ABF-Pre7-4C	1.33	4589.49	0.07	2.65	80.0	3.31	1.34	197.41	0.07	2.62	97.4	2.69	1.33	16.87	0.07	2.64	103.5	2.55	1.33	3.62	0.07	2.65	105.7	2.50
AMG-J	1.49	4793.71	0.10	2.31	81.0	2.85	2.28	1023.37	0.20	1.59	89.2	1.79	2.49	80811.04	0.22	1.49	92.7	1.61	3.04	9463.93	0.27	1.31	94.1	1.39
AMG-4C	1.31	4793.71	0.07	2.69	77.0	3.50	1.78	1023.37	0.14	1.94	84.1	2.31	1.94	80811.04	0.16	1.81	87.4	2.07	2.43	9463.93	0.22	1.52	88.6	1.71
GMG-J	4.62	5220.07	0.37	1.01	83.7	1.20	12.73	619.11	0.56	0.58	92.6	0.62	15.07	95.68	0.59	0.53	96.4	0.55	16.73	15.33	0.61	0.50	97.8	0.51
V(1,1)-4C	1.31	4793.71	0.07	2.69	72.2	3.73	1.78	1023.37	0.14	1.94	79.3	2.45	1.94	80811.04	0.16	1.81	82.6	2.19	2.43	9463.93	0.22	1.52	83.8	1.81
V(0,1)-4C	1.66	4793.71	0.13	2.07	49.2	4.21	3.00	1023.37	0.27	1.32	45.9	2.87	2.37	80811.04	0.21	1.55	46.3	3.35	2.83	9463.93	0.25	1.37	46.7	2.93
CMG	1.72	12088.06	0.13	2.01	42.9	4.68	2.54	850.60	0.23	1.48	43.8	3.37	2.83	76.75	0.25	1.37	46.2	2.96	2.88	11.51	0.26	1.35	48.6	2.78
ABF-Pre7-W	-	4589.49	-	-	80.2	-	-	197.41	-	-	120.2	-	-	16.87	-	-	171.0	-	-	3.62	-	-	240.3	-
ABF-Pre7-4C-W	1.01	4589.49	0.00	6.17	139.9	4.41	1.01	197.41	0.00	6.15	307.5	2.00	1.01	16.87	0.00	6.15	502.1	1.23	1.01	3.62	0.00	6.15	751.2	0.82
AMG-4C-W	1.31	4793.71	0.07	2.69	77.0	3.50	1.39	1023.37	0.08	2.50	110.4	2.27	1.39	80811.04	0.08	2.50	131.4	1.90	1.39	9463.93	0.08	2.51	145.5	1.72

Original condition number =21779.7

(b) theoretical convergence results

Table 125: EPD, 33×33 image

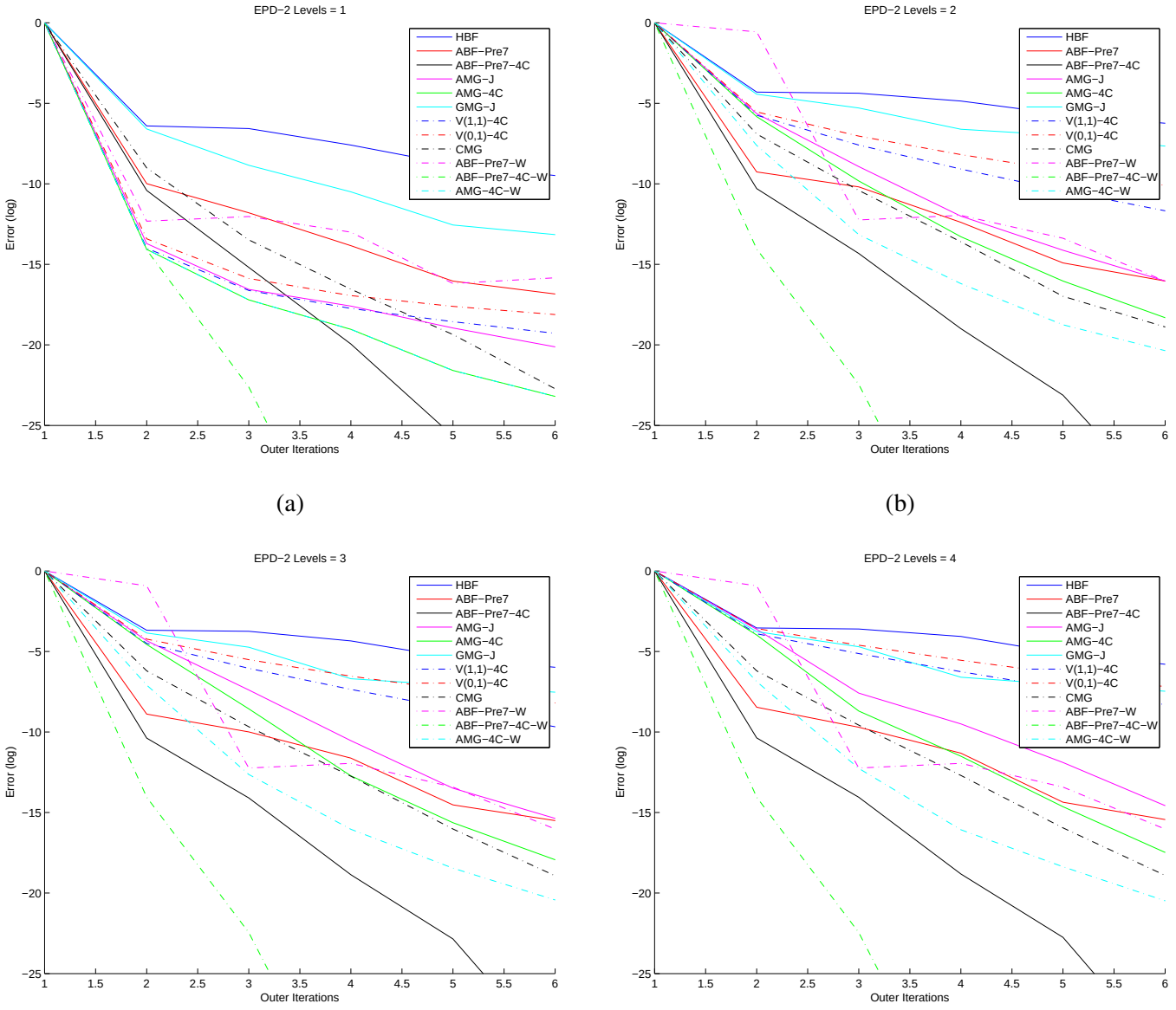


Figure 173: Log error vs. its, EPD, 33×33 image

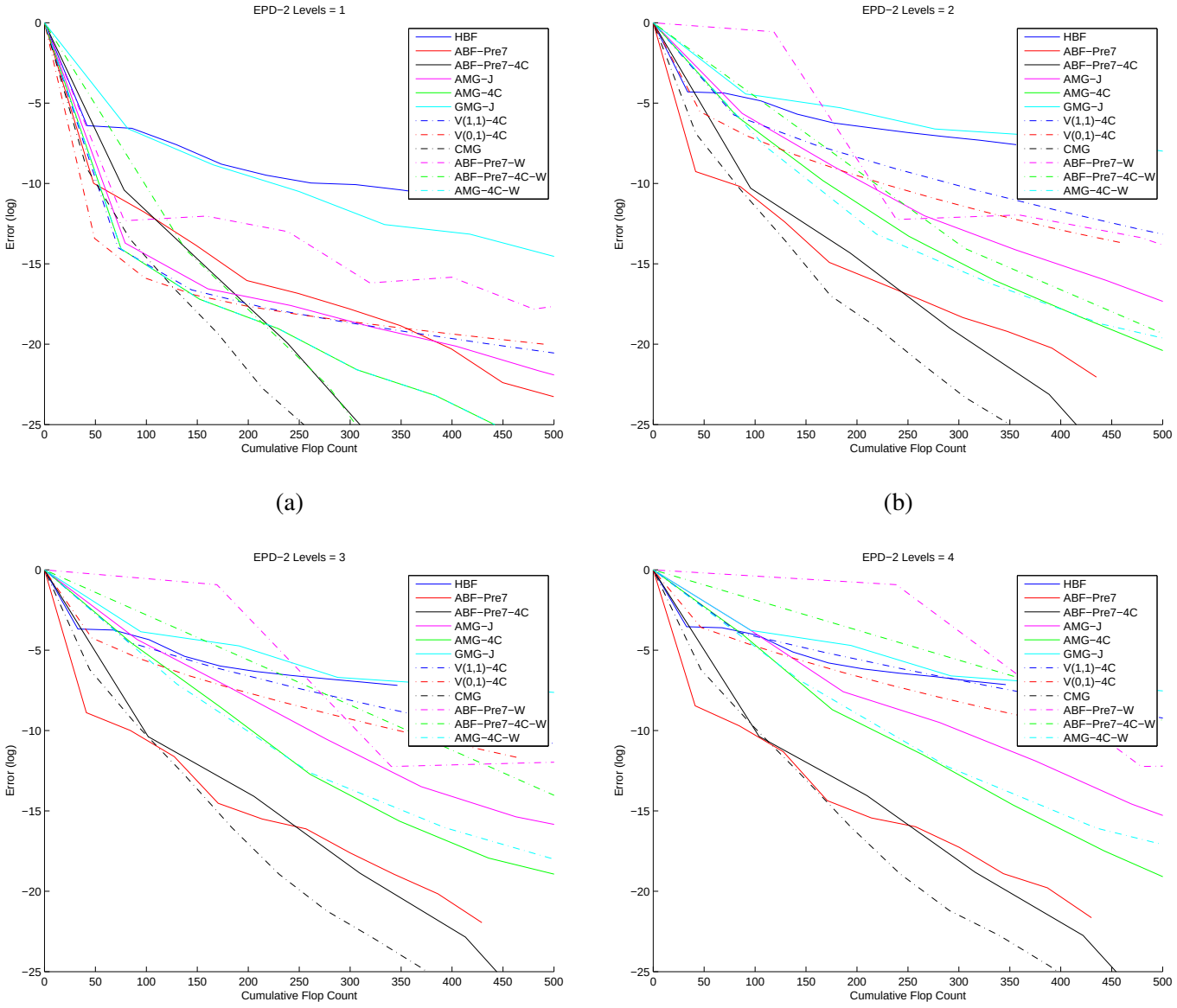


Figure 174: Log error vs. flops, EPD, 33×33 image

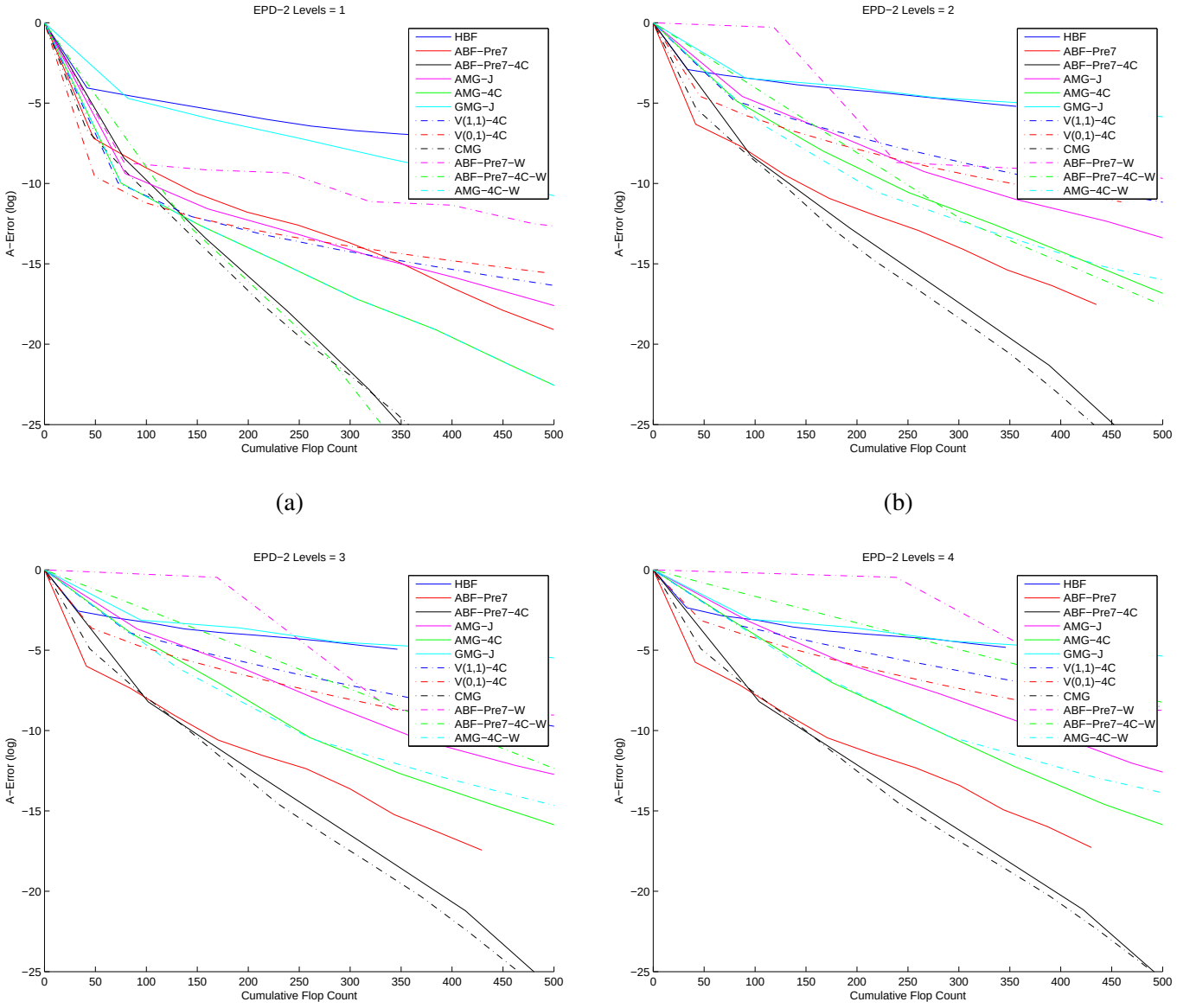


Figure 175: Log A-error vs. flops, EPD, 33×33 image

	$L=5$				$L=6$				$L=7$				$L=8$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.53	0.64	33.2	1.91	0.54	0.62	33.2	1.86	0.55	0.61	33.2	1.83	0.55	0.59	33.2	1.78
ABF-Pre7	0.27	1.29	41.9	3.09	0.27	1.29	41.8	3.09	0.27	1.29	41.8	3.09	0.27	1.29	41.8	3.09
ABF-Pre7-4C	0.10	2.28	101.8	2.24	0.10	2.28	101.8	2.23	0.10	2.28	101.9	2.23	0.10	2.28	101.9	2.23
AMG-J	0.33	1.10	90.8	1.21	0.36	1.01	90.8	1.11	0.43	0.85	90.9	0.94	0.43	0.84	90.9	0.93
AMG-4C	0.27	1.31	85.5	1.53	0.32	1.13	85.5	1.32	0.35	1.04	85.5	1.21	0.36	1.02	85.5	1.19
GMG-J	0.44	0.83	94.5	0.88	0.44	0.83	94.5	0.88	0.43	0.84	94.5	0.89	0.43	0.84	94.5	0.89
V(1,1)-4C	0.48	0.74	80.7	0.91	0.51	0.68	80.7	0.85	0.51	0.67	80.7	0.83	0.51	0.67	80.7	0.82
V(0,1)-4C	0.52	0.66	44.7	1.48	0.53	0.63	44.7	1.40	0.55	0.61	44.7	1.36	0.55	0.60	44.7	1.35
CMG	0.11	2.20	48.9	4.49	0.11	2.19	49.0	4.48	0.11	2.19	49.2	4.46	0.11	2.19	49.4	4.44
ABF-Pre7-W	0.36	1.03	259.8	0.40	0.36	1.03	297.4	0.35	0.36	1.03	343.7	0.30	0.36	1.03	405.5	0.25
ABF-Pre7-4C-W	0.15	1.90	804.8	0.24	0.15	1.90	965.7	0.20	0.15	1.90	1148.2	0.17	0.15	1.90	1374.6	0.14
AMG-4C-W	0.16	1.83	137.7	1.33	0.16	1.82	139.6	1.30	0.16	1.81	140.8	1.28	0.16	1.81	141.6	1.28

(a) empirical convergence results

	$L=5$						$L=6$						$L=7$						$L=8$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	32.86	44.59	0.70	0.35	33.2	1.06	44.33	16.87	0.74	0.30	33.2	0.91	44.96	8.59	0.74	0.30	33.2	0.91	46.16	8.50	0.74	0.30	33.2	0.89
ABF-Pre7	19.25	296.23	0.63	0.46	41.9	1.11	19.26	241.92	0.63	0.46	41.8	1.11	19.26	205.44	0.63	0.46	41.8	1.11	19.26	112.86	0.63	0.46	41.8	1.11
ABF-Pre7-4C	3.50	296.23	0.30	1.19	101.8	1.17	3.50	241.92	0.30	1.19	101.8	1.17	3.50	205.44	0.30	1.19	101.9	1.17	3.50	112.86	0.30	1.19	101.9	1.17
AMG-J	6.28	180799806.28	0.43	0.84	90.8	0.93	6.55	32415954.62	0.44	0.83	90.8	0.91	9.31	30881399.53	0.51	0.68	90.9	0.75	15.32	30083274.93	0.59	0.52	90.9	0.58
AMG-4C	5.79	180799806.28	0.41	0.88	85.5	1.04	5.88	32415954.62	0.42	0.88	85.5	1.03	8.55	30881399.53	0.49	0.71	85.5	0.83	13.64	30083274.93	0.57	0.56	85.5	0.65
GMG-J	8.85	44.59	0.50	0.70	94.5	0.74	9.41	16.87	0.51	0.68	94.5	0.72	10.22	8.59	0.52	0.65	94.5	0.68	10.29	8.50	0.52	0.64	94.5	0.68
V(1,1)-4C	5.79	180799806.28	0.41	0.88	80.7	1.10	5.88	32415954.62	0.42	0.88	80.7	1.09	8.55	30881399.53	0.49	0.71	80.7	0.88	13.64	30083274.93	0.57	0.56	80.7	0.69
V(0,1)-4C	7.28	180799806.28	0.46	0.78	44.7	1.74	7.67	32415954.62	0.47	0.76	44.7	1.69	10.85	30881399.53	0.53	0.63	44.7	1.40	17.33	30083274.93	0.61	0.49	44.7	1.10
CMG	1.09	347.19	0.02	3.79	48.9	7.74	1.10	192.54	0.02	3.72	49.0	7.60	1.10	76.45	0.02	3.71	49.2	7.54	1.10	25.94	0.02	3.71	49.4	7.50
ABF-Pre7-W	-	-	-	-	259.8	-	-	-	-	-	297.4	-	-	-	-	-	343.7	-	-	-	-	-	405.5	-
ABF-Pre7-4C-W	-	-	-	-	804.8	-	-	-	-	-	965.7	-	-	-	-	-	1148.2	-	-	-	-	-	1374.6	-
AMG-4C-W	-	-	-	-	137.7	-	-	-	-	-	139.6	-	-	-	-	-	140.8	-	-	-	-	-	141.6	-

Original condition number =60900.7

(b) theoretical convergence results

Table 126: EPD, 800×530 image

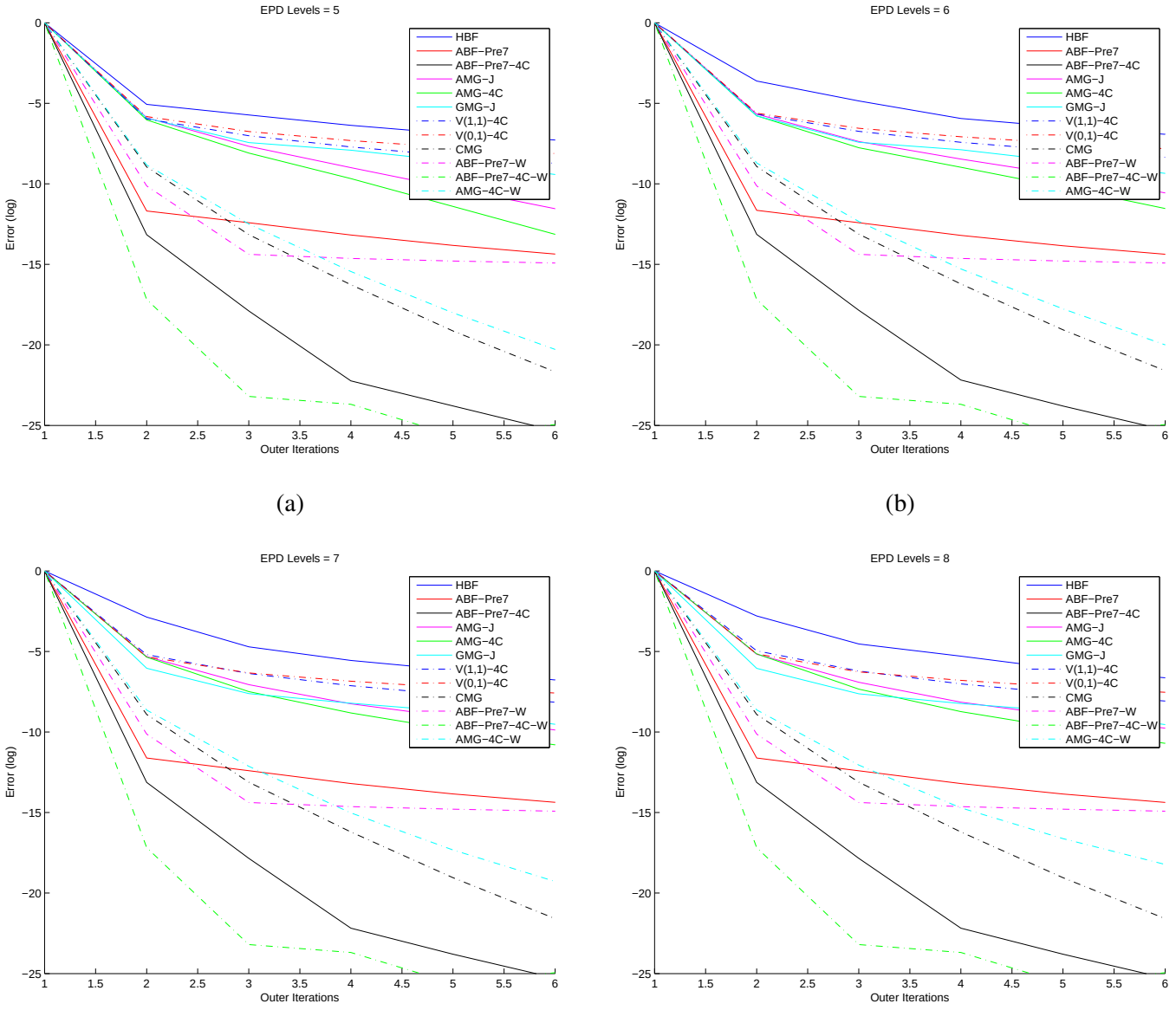


Figure 176: Log error vs. its, EPD, 800×530 image

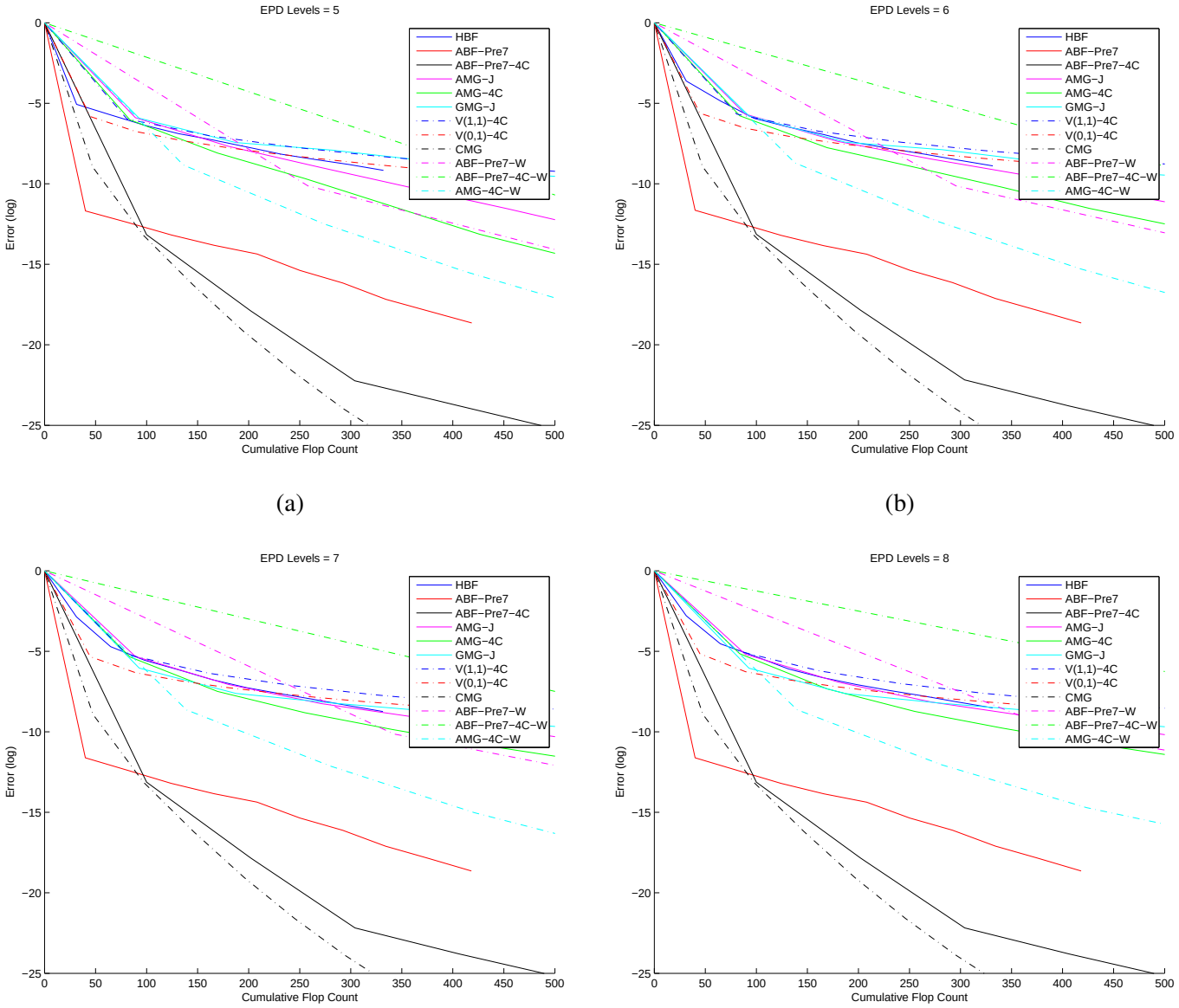
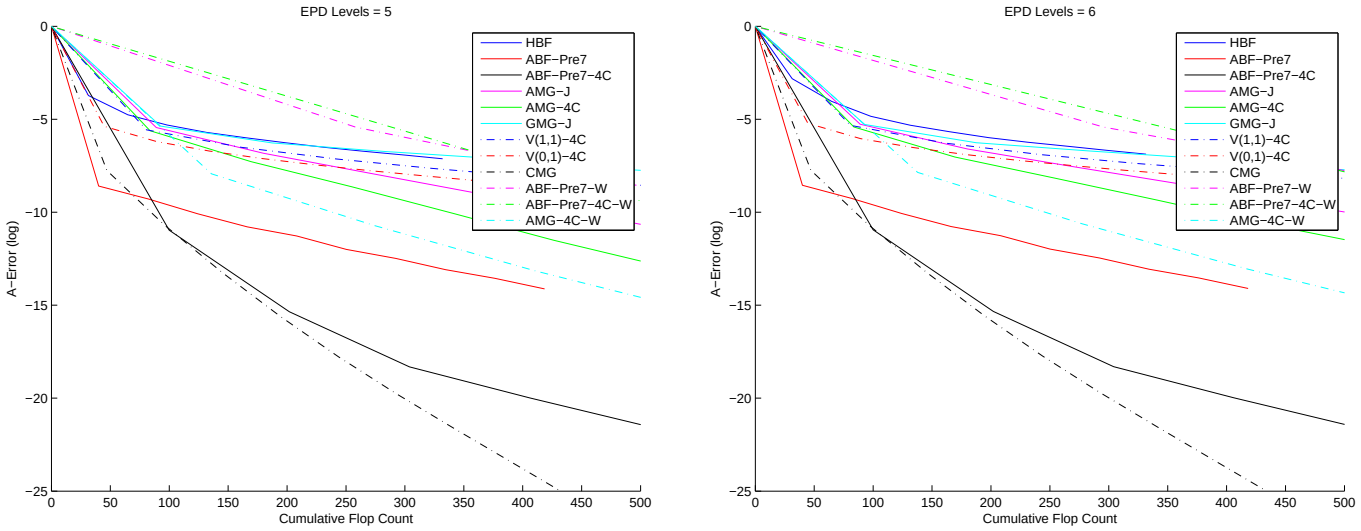
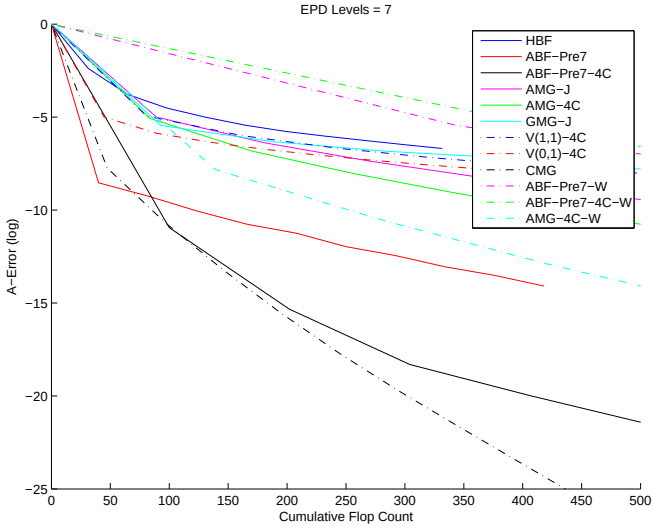


Figure 177: Log error vs. flops, EPD, 800×530 image

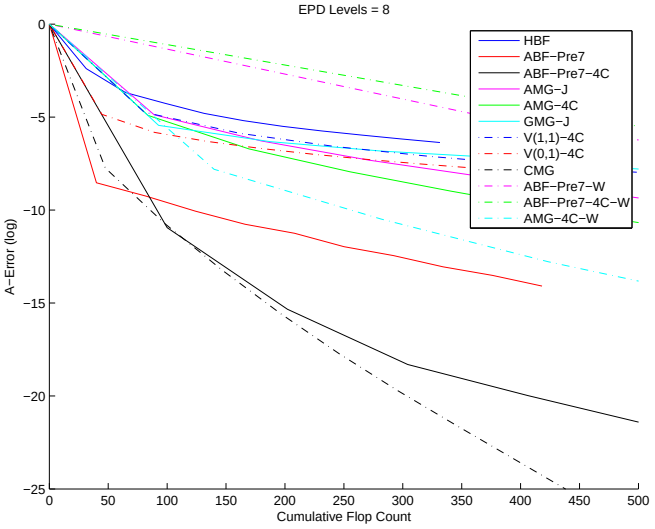


(a)

(b)



(c)



(d)

Figure 178: Log A-error vs. flops, EPD, 800×530 image

	$L=4$				$L=5$				$L=6$				$L=7$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.62	0.48	33.4	1.43	0.63	0.47	33.2	1.41	0.63	0.46	33.2	1.38	0.64	0.45	33.2	1.37
ABF-Pre7	0.34	1.07	42.1	2.54	0.35	1.06	41.9	2.54	0.35	1.06	41.8	2.54	0.35	1.06	41.8	2.54
ABF-Pre7-4C	0.14	1.97	101.7	1.94	0.14	1.97	101.8	1.94	0.14	1.97	101.9	1.93	0.14	1.97	101.9	1.93
AMG-J	0.38	0.96	90.8	1.05	0.46	0.77	90.9	0.85	0.50	0.69	90.9	0.76	0.52	0.66	90.9	0.72
AMG-4C	0.33	1.11	85.5	1.30	0.42	0.87	85.5	1.02	0.45	0.81	85.6	0.94	0.46	0.79	85.6	0.92
GMG-J	0.51	0.67	94.5	0.71	0.51	0.67	94.5	0.71	0.51	0.67	94.6	0.71	0.51	0.67	94.6	0.71
V(1,1)-4C	0.57	0.57	80.7	0.71	0.65	0.43	80.7	0.53	0.70	0.35	80.8	0.44	0.73	0.31	80.8	0.39
V(0,1)-4C	0.59	0.52	44.9	1.16	0.67	0.39	44.7	0.88	0.73	0.32	44.7	0.71	0.76	0.28	44.7	0.62
CMG	0.12	2.09	49.0	4.27	0.12	2.08	48.7	4.27	0.12	2.08	48.8	4.26	0.13	2.08	49.0	4.24
ABF-Pre7-W	0.41	0.90	227.3	0.39	0.41	0.90	259.3	0.35	0.41	0.90	296.7	0.30	0.41	0.90	342.6	0.26
ABF-Pre7-4C-W	0.18	1.70	648.9	0.26	0.18	1.70	803.3	0.21	0.18	1.70	963.3	0.18	0.18	1.70	1144.8	0.15
AMG-4C-W	0.19	1.65	134.4	1.23	0.19	1.64	137.7	1.19	0.20	1.63	139.6	1.17	0.20	1.63	140.8	1.16

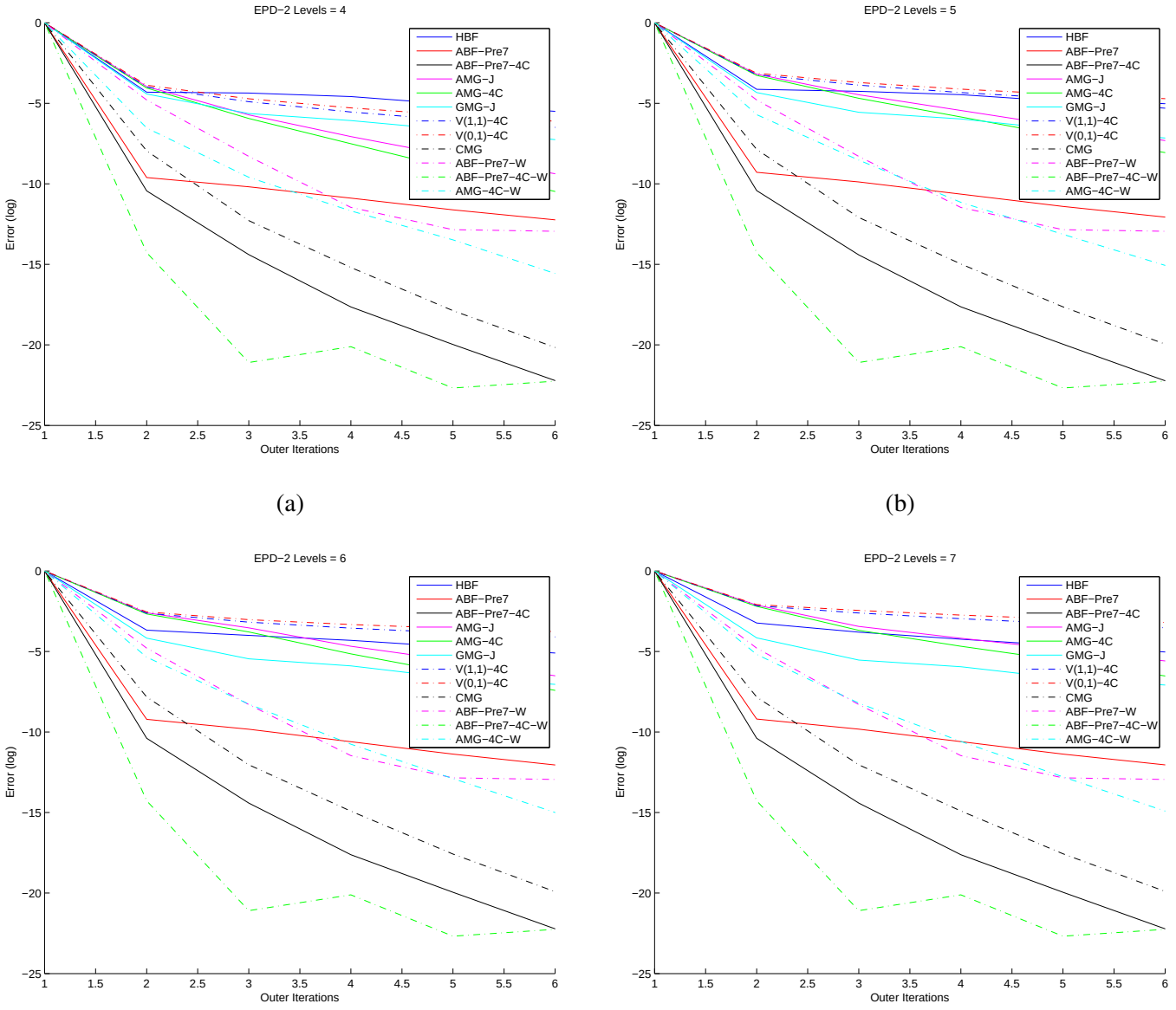
(a) empirical convergence results

	$L=4$						$L=5$						$L=6$						$L=7$					
Algorithm	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$	κ	ν	ρ	τ	C	$\hat{\tau}$
HBF	10.32	69.92	0.53	0.64	33.4	1.93	18.22	17.77	0.62	0.48	33.2	1.44	38.35	8.05	0.72	0.33	33.2	0.98	46.97	5.82	0.75	0.29	33.2	0.89
ABF-Pre7	17.26	261.28	0.61	0.49	42.1	1.17	17.30	119.19	0.61	0.49	41.9	1.17	17.31	68.44	0.61	0.49	41.8	1.17	17.32	49.88	0.61	0.49	41.8	1.17
ABF-Pre7-4C	2.58	261.28	0.23	1.46	101.7	1.43	2.58	119.19	0.23	1.46	101.8	1.43	2.57	68.44	0.23	1.46	101.9	1.43	2.57	49.88	0.23	1.46	101.9	1.43
AMG-J	3.50	650532668.92	0.30	1.19	90.8	1.31	5.29	210795625.67	0.39	0.93	90.9	1.03	7.62	75658.65	0.47	0.76	90.9	0.83	5.91	17500.32	0.42	0.87	90.9	0.96
AMG-4C	3.11	650532668.92	0.28	1.29	85.5	1.50	4.83	210795625.67	0.37	0.98	85.5	1.15	7.03	75658.65	0.45	0.79	85.6	0.93	5.56	17500.32	0.40	0.91	85.6	1.06
GMG-J	6.55	69.92	0.44	0.83	94.5	0.87	7.34	17.77	0.46	0.77	94.5	0.82	8.16	8.05	0.48	0.73	94.6	0.77	8.69	5.82	0.49	0.71	94.6	0.75
V(1,1)-4C	3.11	650532668.92	0.28	1.29	80.7	1.59	4.83	210795625.67	0.37	0.98	80.7	1.22	7.03	75658.65	0.45	0.79	80.8	0.98	5.56	17500.32	0.40	0.91	80.8	1.12
V(0,1)-4C	4.05	650532668.92	0.34	1.09	44.9	2.43	6.13	210795625.67	0.42	0.86	44.7	1.91	8.89	75658.65	0.50	0.70	44.7	1.56	6.61	17500.32	0.44	0.82	44.7	1.84
CMG	1.09	941.36	0.02	3.86	49.0	7.87	1.10	263.84	0.02	3.76	48.7	7.72	1.10	125.55	0.02	3.71	48.8	7.60	1.11	67.91	0.03	3.69	49.0	7.52
ABF-Pre7-W	-	-	-	-	227.3	-	-	-	-	-	259.3	-	-	-	-	-	296.7	-	-	-	-	-	342.6	-
ABF-Pre7-4C-W	-	-	-	-	648.9	-	-	-	-	-	803.3	-	-	-	-	-	963.3	-	-	-	-	-	1144.8	-
AMG-4C-W	-	-	-	-	134.4	-	-	-	-	-	137.7	-	-	-	-	-	139.6	-	-	-	-	-	140.8	-

Original condition number =61050.9

(b) theoretical convergence results

Table 127: EPD, 535×802 image



(a) (b)

(c) (d)

Figure 179: Log error vs. its, EPD, 535×802 image

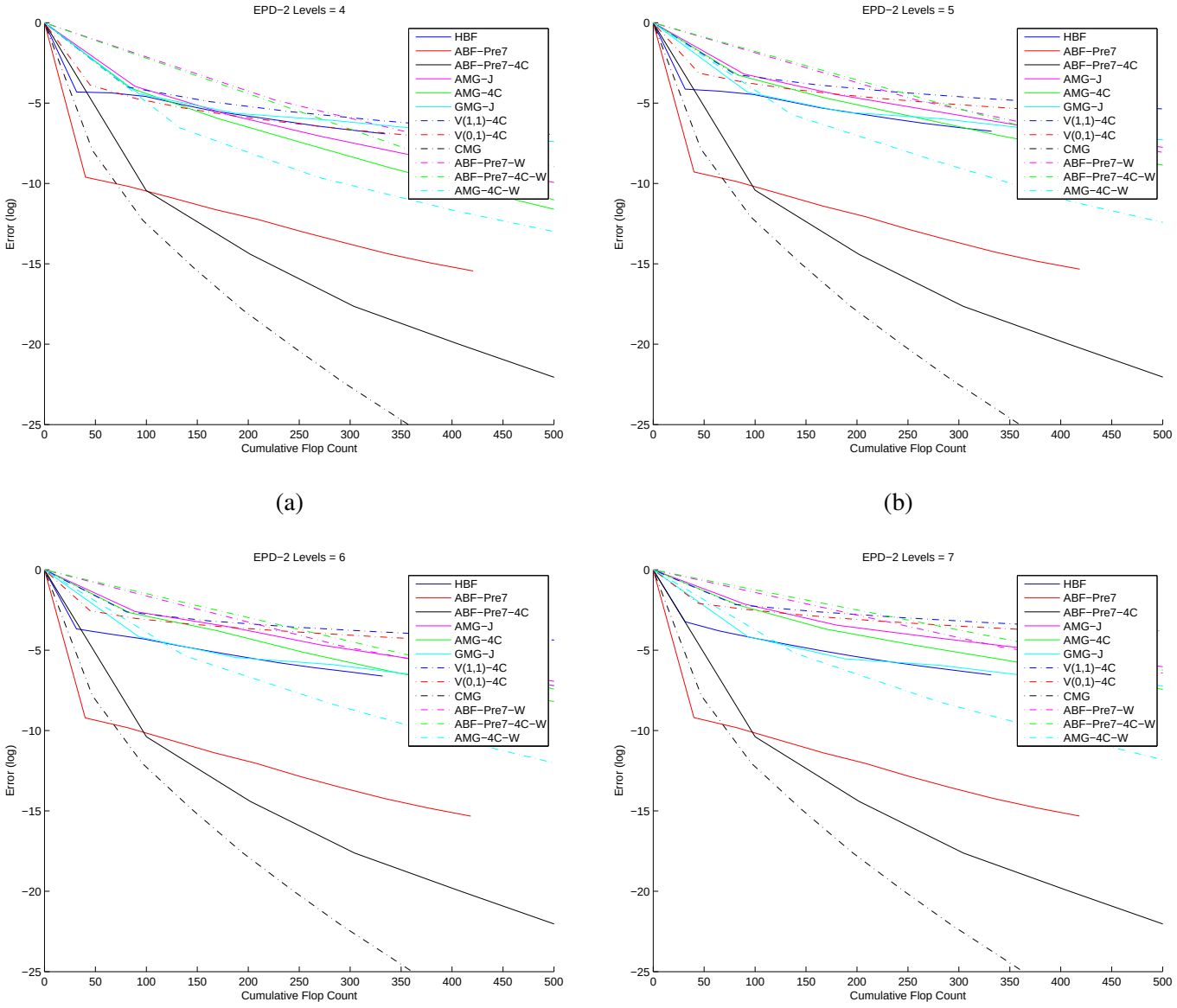
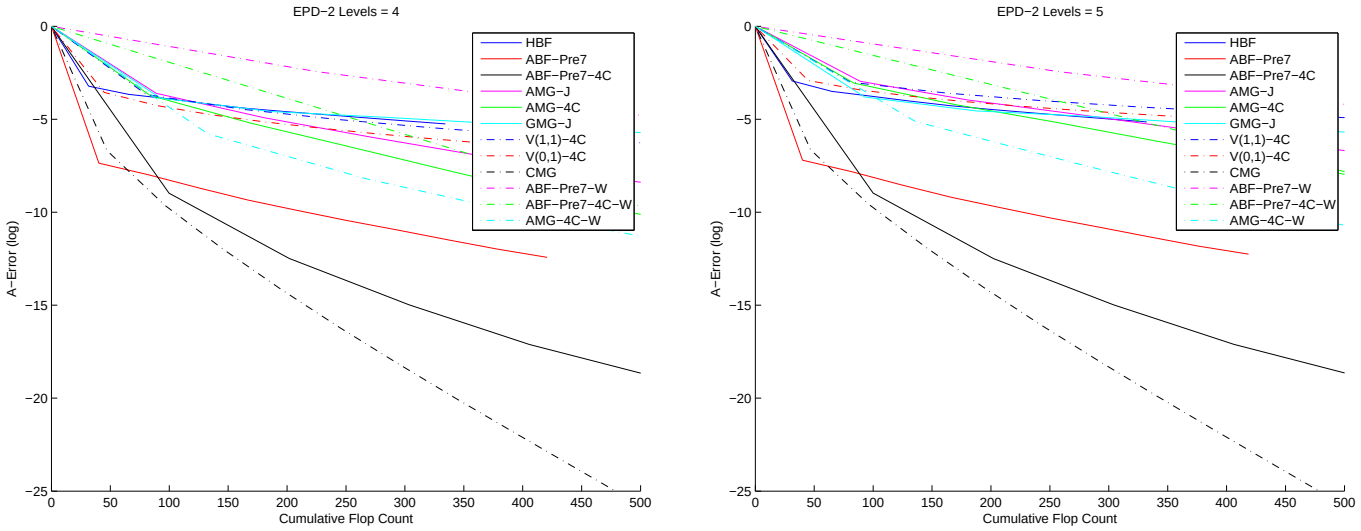
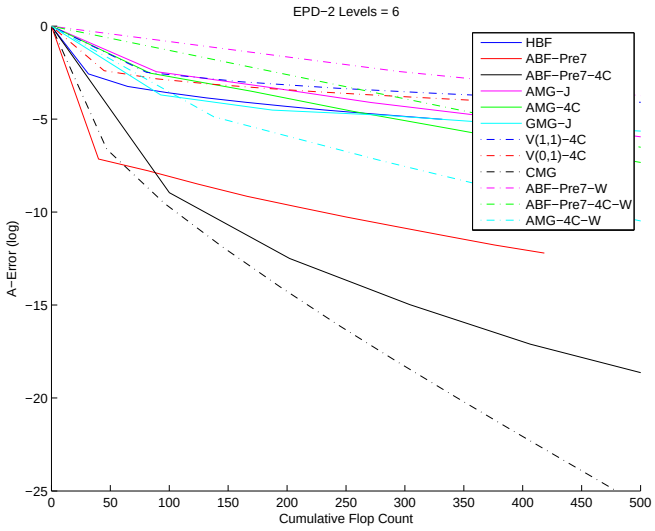


Figure 180: Log error vs. flops, EPD, 535×802 image

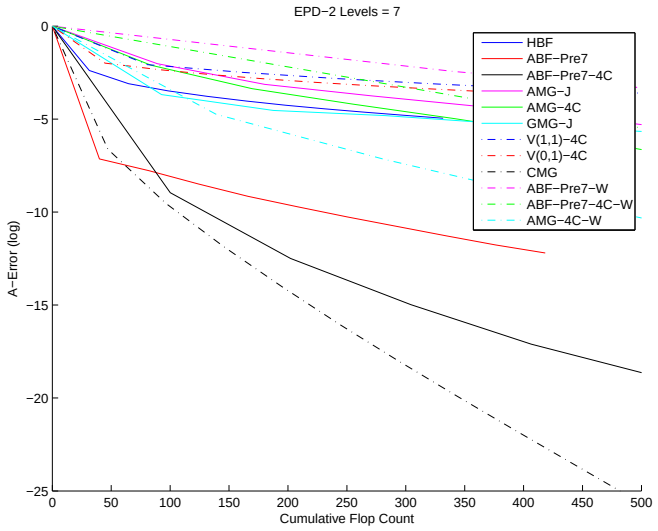


(a)

(b)



(c)



(d)

Figure 181: Log A-error vs. flops, EPD, 535×802 image

	$L=5$				$L=6$				$L=7$				$L=8$			
Algorithm	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$	$\tilde{\rho}$	$\tilde{\tau}$	C	$\hat{\tau}$
HBF	0.55	0.59	33.2	1.78	0.57	0.57	33.2	1.72	0.57	0.55	33.2	1.67	0.58	0.55	33.1	1.65
ABF-Pre7	0.34	1.09	41.9	2.60	0.34	1.09	41.8	2.60	0.34	1.09	41.8	2.60	0.34	1.09	41.8	2.60
ABF-Pre7-4C	0.17	1.77	101.8	1.74	0.17	1.77	101.8	1.74	0.17	1.77	101.8	1.74	0.17	1.77	101.8	1.74
AMG-J	0.48	0.73	90.8	0.80	0.54	0.62	90.8	0.68	0.58	0.55	90.8	0.60	0.58	0.55	90.8	0.60
AMG-4C	0.40	0.92	85.5	1.07	0.44	0.82	85.5	0.96	0.47	0.76	85.5	0.89	0.47	0.75	85.5	0.87
GMG-J	0.42	0.86	94.5	0.91	0.42	0.86	94.5	0.91	0.42	0.86	94.5	0.91	0.42	0.86	94.5	0.91
V(1,1)-4C	0.62	0.47	80.7	0.59	0.69	0.37	80.7	0.46	0.73	0.32	80.7	0.40	0.74	0.30	80.7	0.37
V(0,1)-4C	0.64	0.44	44.7	0.98	0.71	0.34	44.7	0.76	0.75	0.29	44.7	0.64	0.77	0.26	44.7	0.59
CMG	0.13	2.02	48.3	4.18	0.13	2.01	48.1	4.18	0.13	2.01	48.1	4.18	0.13	2.01	48.2	4.18

(a) empirical convergence results

Table 128: EPD, 1365×2048 image

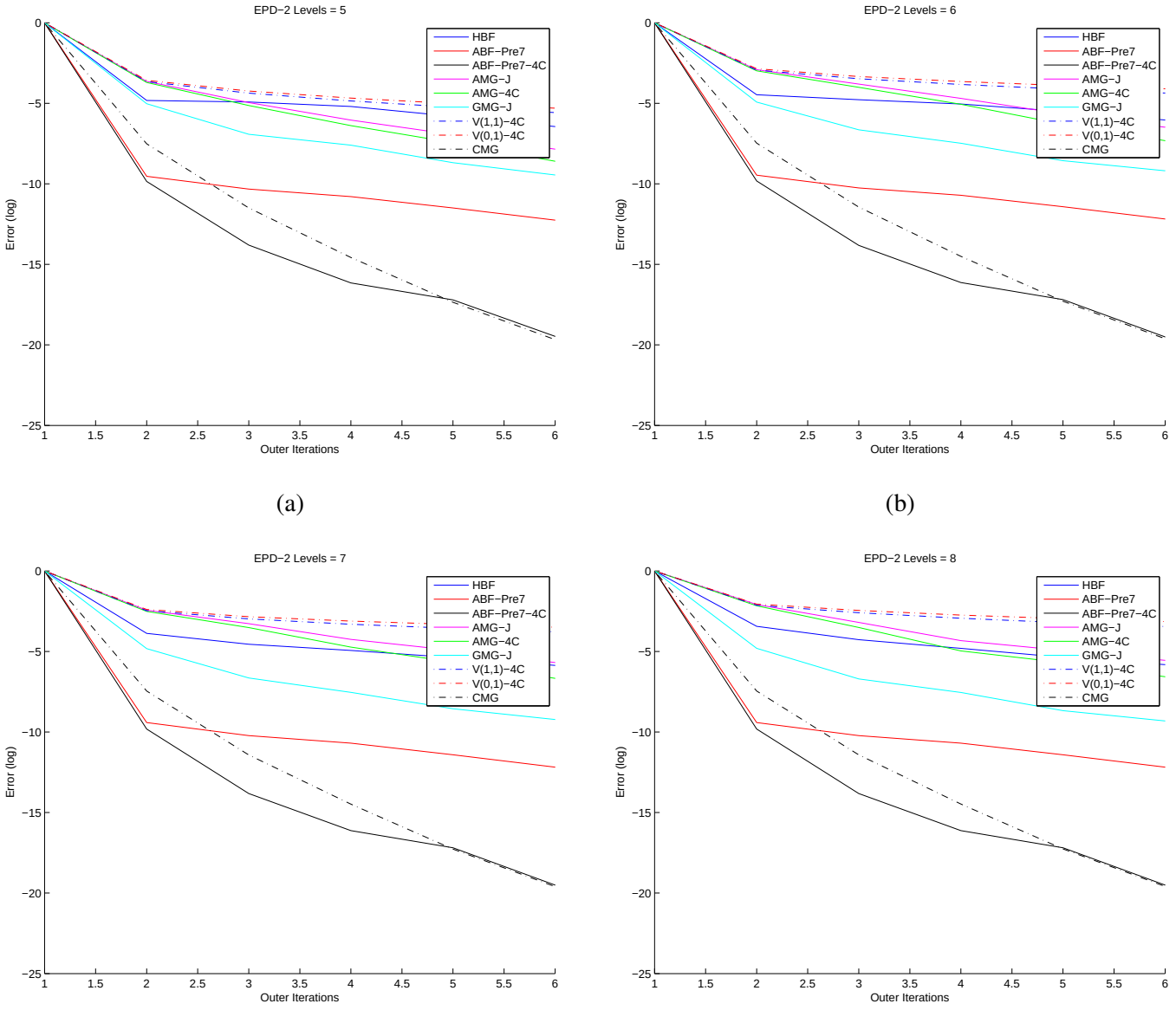
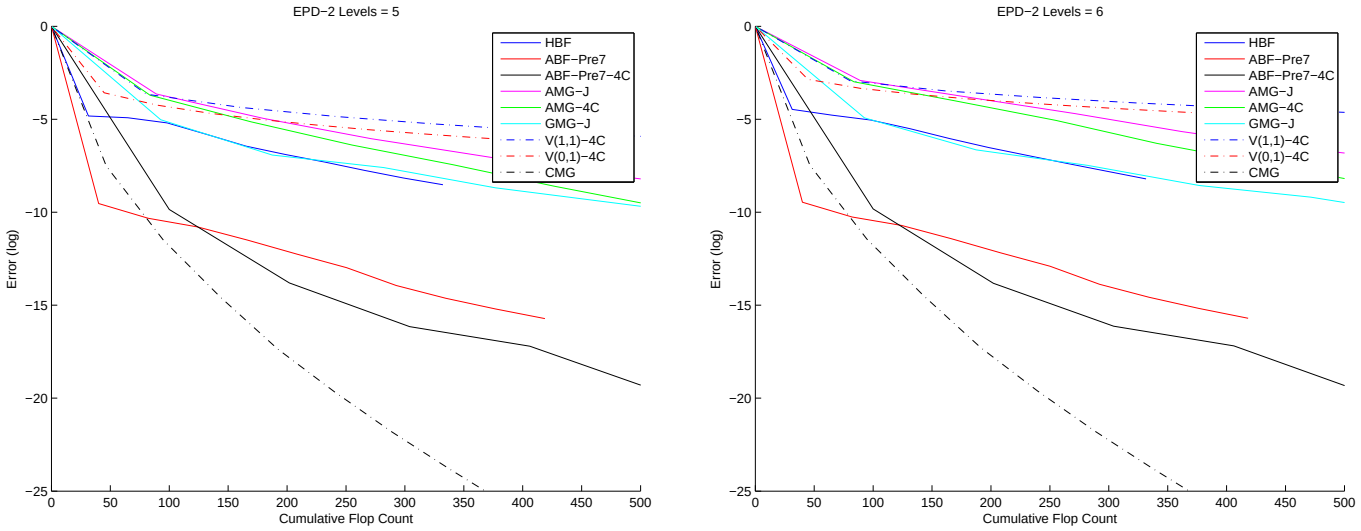
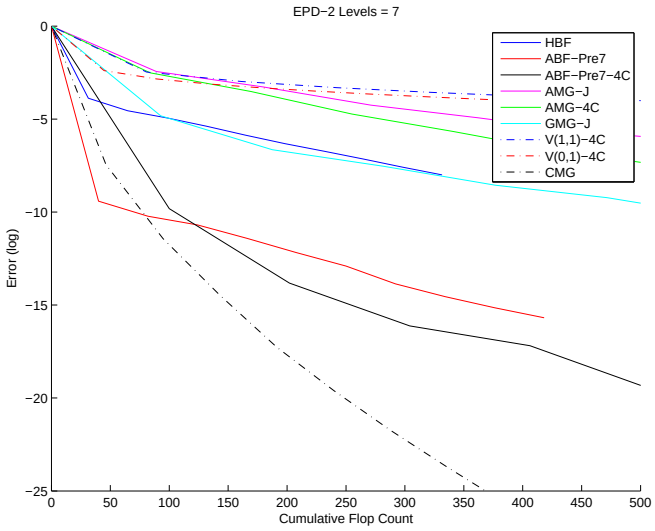


Figure 182: Log error vs. its, EPD, 1365×2048 image

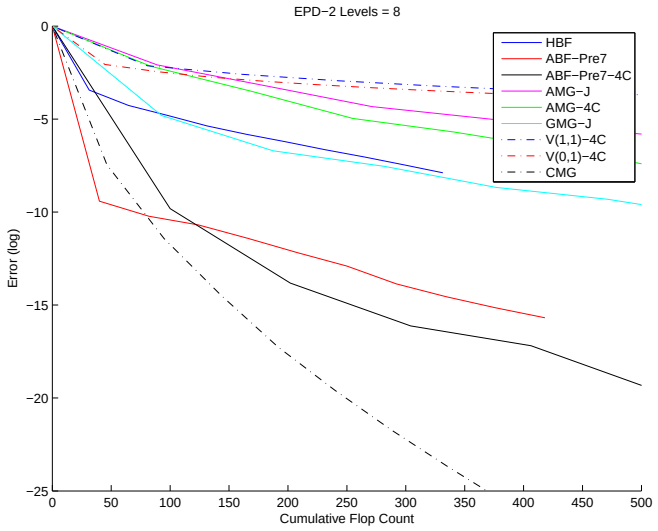


(a)

(b)

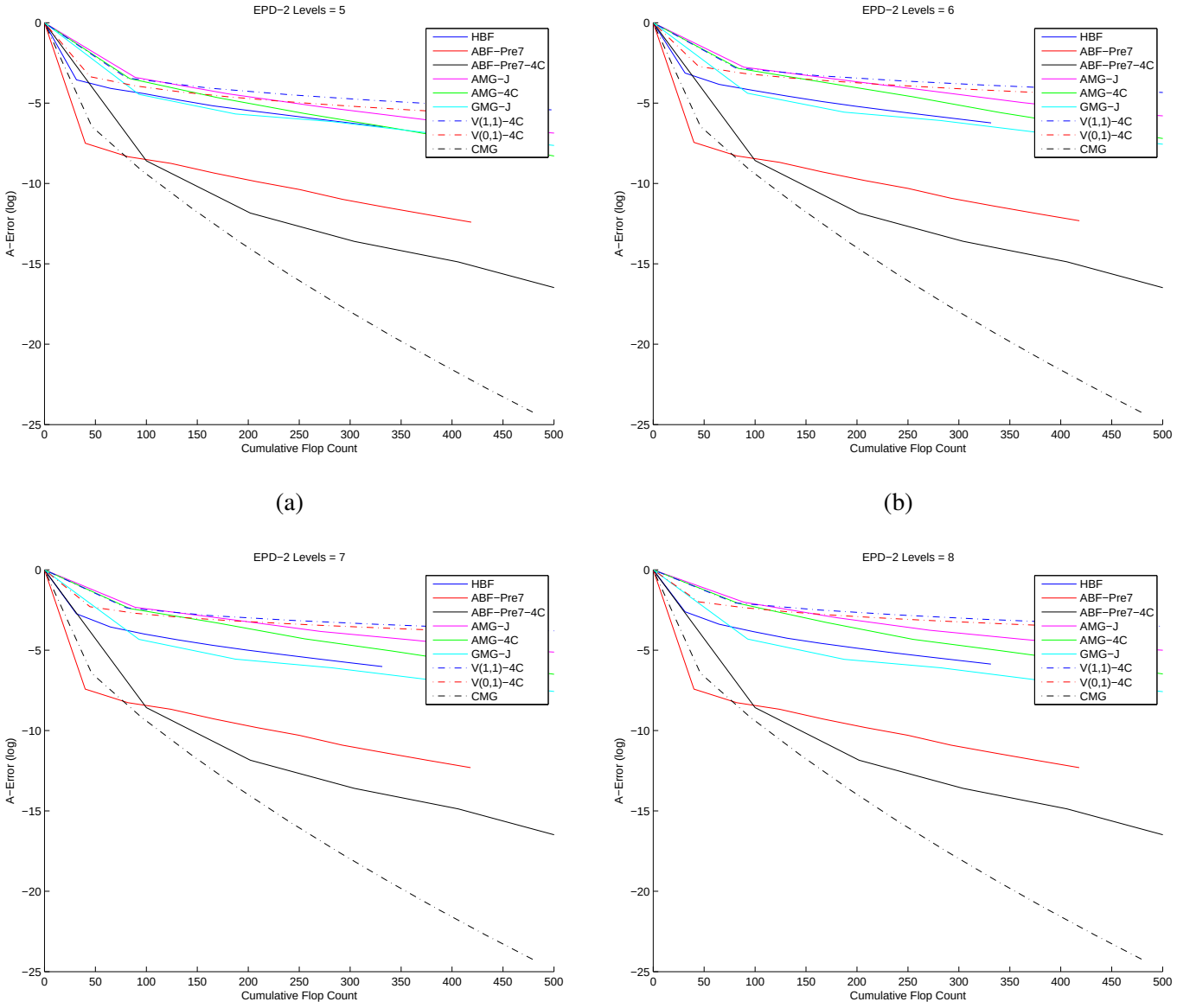


(c)



(d)

Figure 183: Log error vs. flops, EPD, 1365×2048 image



(a) (b)

(c) (d)

Figure 184: Log A-error vs. flops, EPD, 1365×2048 image