Parallel Total Variation Minimization

Jahn Müller

jahn.mueller@uni-muenster.de

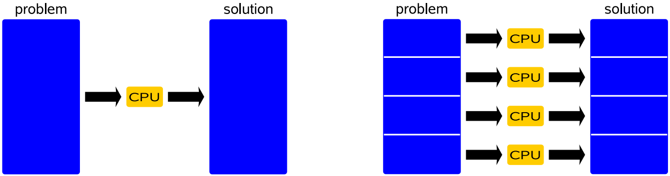
Westfälische Wilhelms-Universität Münster

27.01.2009

Outline

Introduction

- desirable to extend 2D algorithms to 3D
- currently used workstations reach their technical limitations
- expedient:
  - divide original problem in subproblems
  - solve independently on several CPU's
  - merge together to a solution of the complete problem



- data dependences may arise (neglecting leads to undesirable effects at the interfaces)

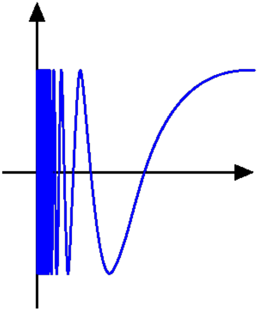
Image Denoising

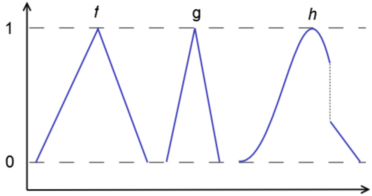
function space for denoising

- Lebesgue space
  $L^p(\Omega) := \{u : \Omega \to \bar{\mathbb{R}} \mid \int_\Omega |u|^p dx < \infty\}, \; p \geq 1$
  - contain oscillating images, in particular noise
- Sobolev space
  $W^{1,p}(\Omega) := \{u \in L^p(\Omega) \mid \frac{\partial u}{\partial x_j} \in L^p(\Omega), j = 1, \ldots, d\}, \; p \geq 1$
  - to restrictiv, e.g. for piecewise constant images

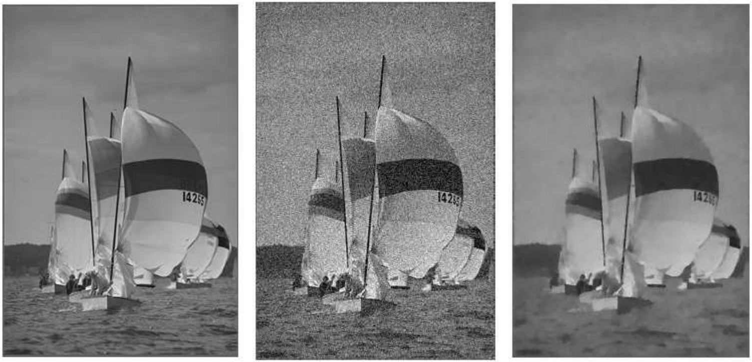
- need for a proper space
- space of functions with bounded variation (BV)

Total Variation





Total Variation Denoising



Total Variation Regularization

Let $f : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ be a noisy version of a given image u_0 with noise variance given by $\int_\Omega (u_0 - f)^2 \, dx \leq \sigma^2$.

The ROF model: (Rudin, Osher, Fatemi)

$$\hat{u} = \underset{u \in BV(\Omega)}{\arg\min} \left\{ \underbrace{\frac{\lambda}{2} \int_\Omega (u - f)^2 \, dx}_{\text{data fitting}} + \underbrace{|u|_{TV}}_{\text{regularization}} \right\} \tag{1}$$

where

- $|u|_{TV} := \sup_{\substack{\varphi \in \mathcal{C}_0^\infty(\Omega)^d \\ ||\varphi||_\infty \leq 1}} \int_\Omega u \nabla \cdot \varphi \, dx$
  denotes the total variation of u
- $BV(\Omega) = \left\{ u \in L^1(\Omega) \mid |u|_{TV} < \infty \right\}$ is the space of functions with bounded total variation.

Total Variation Regularization

- depending on exact definition of the supremum norm

$$||p||_\infty := \operatorname*{ess\,sup}_{x\in\Omega} ||p(x)||_{l^r}$$

family of equivalent seminorms:

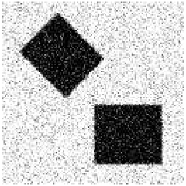
$$\int_\Omega |Du|_{l^s} = \sup_{\substack{\varphi\in\mathcal{C}_0^\infty(\Omega)^d \\ ||\varphi||_\infty\leq 1}} \int_\Omega u\nabla\cdot\varphi\,dx$$

with $\frac{1}{s}+\frac{1}{r}=1$ (Hölder conjugate)

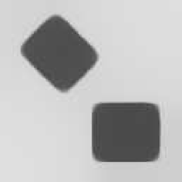
- isotropic total variation ($r=2$)
- anisotropic total variation ($r=\infty$)

Isotropic vs Anistropic Total Variation

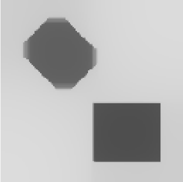
(a) Original image



(b) Noisy image



(c) Isotropic TV



(d) Anisotropic TV

Total Variation Regularization

- The functional

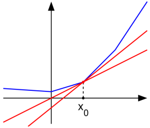
$$J(u) := \frac{\lambda}{2}\int_\Omega (u-f)^2\,dx + \int_\Omega |u|_{TV}$$

is strict convex and attains a unique minimum in BV (lower semicontinuity and weak*- compactness of the sub-level sets).

- optimality condition in terms of subgradients $(\partial J(u) := \{p \in \mathcal{U}^* \mid J(w) \geq J(u) + \langle p, w-u \rangle, \forall w \in \mathcal{U}\})$:

$$0 \in \partial J(u)$$

$$0 \in \lambda(u-f) + \partial|u|_{TV}$$



Total Variation Regularization

Primal Formulation:

$$J(u) = \frac{\lambda}{2} \int_\Omega (u - f)^2 \, dx + \int_\Omega |\nabla u| \, dx$$

for u sufficiently smooth, particularly $u \in W^{1,1}(\Omega)$.

- Euler Lagrange equation:

$$\lambda(u - f) - \nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right) = 0$$

- perturbe norm to overcome issue with singularity at $\nabla u = 0$:

$$|\nabla u|_\beta := \sqrt{|\nabla u|^2 + \beta}$$

- solution methods: steepest descent (Rudin et al.), fixed point iteration (Vogel and Oman), Newton's method (Chan et al.)

Total Variation Regularization

- exact formulation:

$$\min_{u \in BV(\Omega)} \sup_{\|p\|_\infty \leq 1} \underbrace{\left[\frac{\lambda}{2} \int_\Omega (u-f)^2 \, dx + \int_\Omega u \nabla \cdot p \, dx \right]}_{=:L(u,p)}$$

- interchange min and sup (not trivial!):

$$\min_u \sup_p L(u,p) = \max_p \min_u L(u,p)$$

Dual Formulation:

$$\min_{\|p\|_\infty \leq 1} \left\| \frac{1}{\lambda} \nabla \cdot p - f \right\|_2^2 \tag{2}$$

- after solving (2) we obtain the solution for u from $u = f - \frac{1}{\lambda} \nabla \cdot p$
- solution methods: projection algorithm (Chambolle)

Total Variation Regularization

Primal Dual Formulation:

$$\inf_{u \in BV(\Omega)} \sup_{\|p\|_\infty \leq 1} \underbrace{\left[\frac{\lambda}{2} \int_\Omega (u-f)^2 \, dx + \int_\Omega u \nabla \cdot p \, dx \right]}_{=:L(u,p)}$$

- For a saddle point we achieve the following optimality conditions

$$\frac{\partial L}{\partial u} = \lambda(u-f) + \nabla \cdot p = 0 \tag{3}$$

  and

$$L(u,p) \geq L(u,q) \qquad \forall q \,,\, \|q\|_\infty \leq 1, \tag{4}$$

- (4) implies $\nabla \cdot p \in \partial |u|_{TV}$ and hence $0 \in \partial J(u)$

Penalty and Barrier Approximation

- approximate the constraint $\|p\| \leq 1$ in (3) by using Barrier or Penalty methods

Penalty approximation

$$L_\varepsilon(u, p) = L(u, p) - \frac{1}{\varepsilon} F(\|p\| - 1)$$

with $\varepsilon > 0$ small and a term F penalizing if $\|p\| - 1 > 0$.
Typical example:

$$F(s) = \frac{1}{2} \max\{s, 0\}^2$$

- still allows violations of the constraint

Penalty and Barrier Approximation

Barrier approximation

add a continuous barrier term to L such that $G(s) = \infty$, if the constraint is violated

$$L_\varepsilon(u, p) = L(u, p) - \varepsilon G(\|p\|^2 - 1)$$

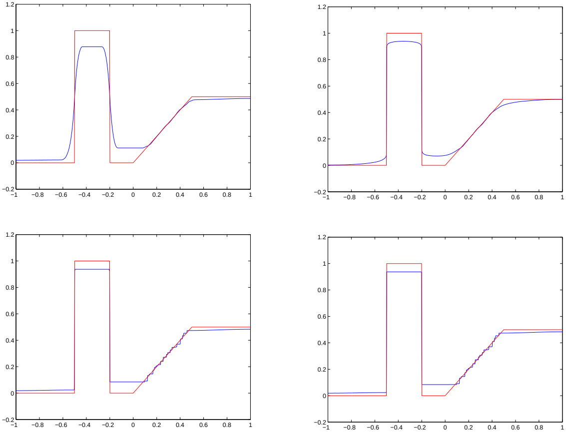
For example:

$$G(s) = -\log(-s)$$

- ensures that the constraint is not violated
- also called interior-point method

- choice of approximation effects the shape of the solution u

Penalty and Barrier Approximation



$$h = 10^{-3}$$
$$\varepsilon = 0.1$$

$$h = 10^{-3}$$
$$\varepsilon = 10^{-3}$$

Penalty method

Barrier method

Newton method with damping

- Optimality conditions (using penalty approximation)

$$\begin{aligned} \frac{\partial L_\varepsilon}{\partial u} &= \quad \lambda(u-f)+\nabla\cdot p &= 0 \\ \frac{\partial L_\varepsilon}{\partial p} &= \quad -\nabla u - \tfrac{2}{\varepsilon}H(p) &= 0 \end{aligned}$$

  with $H(p)$ being the derivative of $F(\|p\|-1)$.
- Linearize $H(p)$ via first-order Taylor-approximation

$$H(p^{k+1}) \approx H(p^k)+H'(p^k)(p^{k+1}-p^k)$$

- Add damping term, with damping parameter τ

$$\tau^k(p^{k+1}-p^k)$$

Newton method with damping

Solve in each step

$$\lambda(u^{k+1} - f) + \nabla \cdot p^{k+1} = 0$$

$$-\nabla u^{k+1} - \tfrac{2}{\varepsilon} H'(p^k)(p^{k+1} - p^k) - \tfrac{2}{\varepsilon} H(p^k) - \tau^k(p^{k+1} - p^k) = 0$$

- linear system and easy to descretize
- choose $\varepsilon \to 0$ during iteration, to obtain fast convergence
- start with small value τ and increase during iteration, to avoid oscillations
- ε and τ are chosen from experimental runs

back

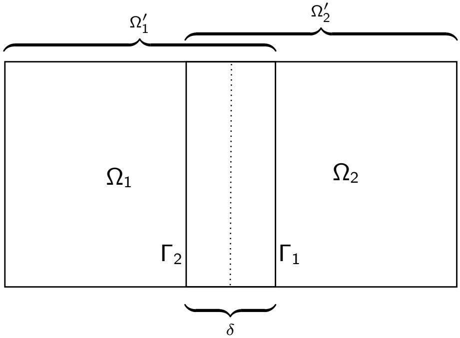
Domain Decomposition

Example

$$\begin{aligned} Lu &= f && \text{in } \Omega \\ u &= 0 && \text{on } \partial\Omega \end{aligned}$$

- split the given domain Ω of the problem into subdomains Ω_i, $i = 1, \ldots, S$
- overlapping or non overlapping decompositions
- when all unknowns are coupled, a straight forward splitting leads to signifant errors at the interfaces
- transmission conditions at the interface
- avoid computing transmission conditions by using overlapping decompositions

Overlapping Decomposition



- for a uniform lattice with stepsize h, $\delta = mh$, with $m \in \mathbb{N}$
- redundant degress of freedom
- achieve update for boundary data from this redundancy

Additive Schwarz Method

Let $u^{(0)}$ be an initial function,

$$\begin{cases} Lu_1^{(k+1)} = f, & \text{in } \Omega_1 \\ u_1^{(k+1)} = u^{(k)}{}_{|\Gamma_1}, & \text{on } \Gamma_1 \\ u_1^{(k+1)} = 0, & \text{on } \partial\Omega_1 \setminus \Gamma_1 \end{cases} \quad \text{and} \quad \begin{cases} Lu_2^{(k+1)} = f, & \text{in } \Omega_2 \\ u_2^{(k+1)} = u_1^{(k)}{}_{|\Gamma_2}, & \text{on } \Gamma_2 \\ u_2^{(k+1)} = 0, & \text{on } \partial\Omega_2 \setminus \Gamma_2. \end{cases}$$

The next step is computed by

$$u^{(k+1)}(x) = \begin{cases} u_2^{(k+1)}(x), & \text{if } x \in \Omega \setminus \Omega_1 \\ u_1^{(k+1)}(x), & \text{if } x \in \Omega \setminus \Omega_2 \\ \frac{u_1^{(k+1)}(x) + u_2^{(k+1)}(x)}{2} & \text{if } x \in \Omega_1 \cap \Omega_2. \end{cases}$$

- related to the well-known Jacobi method
- direct application of parallelization

Multiplicative Schwarz Method

Let $u^{(0)}$ be an initial function,

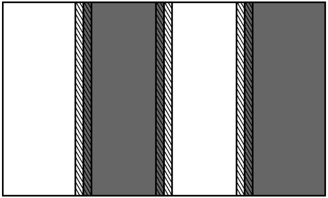
$$\begin{cases} Lu_1^{(k+1)} = f, & \text{in } \Omega_1 \\ u_1^{(k+1)} = u^{(k)}{}_{|\Gamma_1}, & \text{on } \Gamma_1 \\ u_1^{(k+1)} = 0, & \text{on } \partial\Omega_1 \setminus \Gamma_1 \end{cases} \quad \text{and} \quad \begin{cases} Lu_2^{(k+1)} = f, & \text{in } \Omega_2 \\ u_2^{(k+1)} = u_1^{(k+1)}{}_{|\Gamma_2}, & \text{on } \Gamma_2 \\ u_2^{(k+1)} = 0, & \text{on } \partial\Omega_2 \setminus \Gamma_2. \end{cases}$$

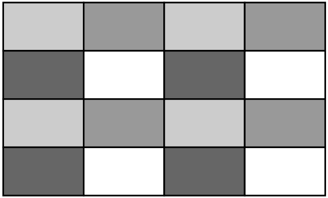
The next step is computed by

$$u^{(k+1)}(x) = \begin{cases} u_2^{(k+1)}(x), & \text{if } x \in \Omega_2 \\ u_1^{(k+1)}(x), & \text{if } x \in \Omega \setminus \Omega_2. \end{cases}$$

- related to the well-known Gauss-Seidel method
- seems not conventient for a parallel implementation
- need of coloring

Multiplicative Schwarz Method - Coloring





- painting subdomins in several colors, such that same colored domains do not overlap
- solve domains of same color in parallel using the latest boundary conditions from the other “colors“

Numerical Realization

$$
\begin{array}{ccccccc}
\bullet u_{11} & * p^2_{11} & \bullet u_{12} & * p^2_{12} & \cdots & * p^2_{1n-1} & \bullet u_{1n} \\
* p^1_{11} & & * p^1_{12} & & \cdots & & * p^1_{1n} \\
\bullet u_{21} & * p^2_{21} & \bullet u_{22} & * p^2_{22} & \cdots & * p^2_{2n-1} & \bullet u_{2n} \\
* p^1_{21} & & * p^1_{22} & & \cdots & & * p^1_{2n} \\
\vdots & & \vdots & & \ddots & & \vdots \\
* p^1_{n-11} & & * p^1_{n-12} & & \cdots & & * p^1_{n-1n} \\
\bullet u_{n1} & * p^2_{n1} & \bullet u_{n2} & * p^2_{n2} & \cdots & * p^2_{nn-1} & \bullet u_{nn}
\end{array}
$$

- Lay the degrees of freedom of the dual variable p in the center between the pixels of u
- compute the divergence of p effective as a value in each pixel (using a single-sided difference quotient)

Numerical Realization

- applying anisotropic total variation
- slightly change penalty term F:

$$F(p) = \frac{1}{2}\max\{|p_1| - 1, 0\}^2 + \frac{1}{2}\max\{|p_2| - 1, 0\}^2$$

and hence

$$H(p) = \begin{pmatrix} \operatorname{sgn}(p_1) \cdot (|p_1| - 1) \cdot \mathbb{1}_{\{|p_1| \geq 1\}} \\ \operatorname{sgn}(p_2) \cdot (|p_2| - 1) \cdot \mathbb{1}_{\{|p_2| \geq 1\}} \end{pmatrix}$$

and its Hessian

$$H'(p) = \begin{pmatrix} \mathbb{1}_{\{|p_1| \geq 1\}} & 0 \\ 0 & \mathbb{1}_{\{|p_2| \geq 1\}} \end{pmatrix}$$

Newton with damping

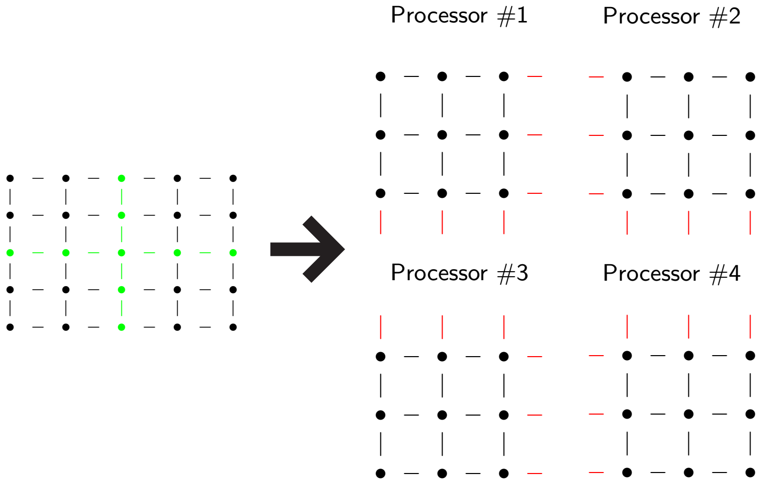
Numerical Realization

- write u, p^1, p^2 column wise in vectors, to obtain a vector $x = (\vec{u}, \vec{p^1}, \vec{p^2})$
- construct a system matrix A and righthand side b to obtain a system $Ax = b$

$$A = \left(\begin{array}{c:c:c} \begin{matrix} \lambda & & \\ & \ddots & \\ & & \lambda \end{matrix} & D_1 & D_2 \\ \hdashline D_1^t & M_1 & 0 \\ \hdashline D_2^t & 0 & M_2 \end{array}\right)$$

- due to the shape of A some improvements to solve this system are applicable , e.g. Schur complement, or conjugate gradient methods.

Parallel Implementation



Parallel Implementation

- “ghost cells“(red) have to be communicated after each iteration
- explicit communication needed
- communication realized via the Message Passing Interface (MPI)
- MatlabMPI makes MPI available in MATLAB (provided by the Lincoln Laboratory of the Massachusetts Institute Of Technology (MIT))

Additive Version

- Neumann boundary conditions for u turn into Dirichlet boundary conditions for p

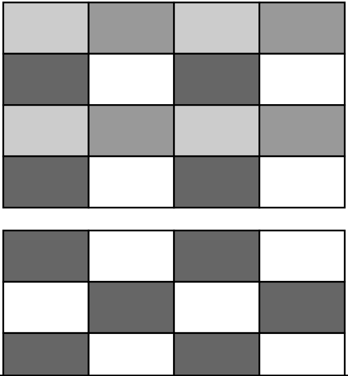
$$\begin{aligned} A\begin{pmatrix} u \\ p \end{pmatrix} &= b && \text{in } \Omega \\ p &= 0 && \text{on } \partial\Omega. \end{aligned}$$

- solve in each step

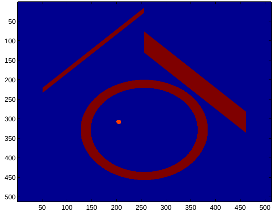
$$\begin{aligned} A_{|\Omega_1}\begin{pmatrix} u_1^{(k+1)} \\ p_1^{(k+1)} \end{pmatrix} &= b_{|\Omega_1} && \text{in } \Omega_1 \\ p_1^{(k+1)} &= p_2^{(k)} && \text{on } \Gamma_1 \\ p_1^{(k+1)} &= 0 && \text{on } \partial\Omega_1 \setminus \Gamma_1 \end{aligned} \qquad \begin{aligned} A_{|\Omega_2}\begin{pmatrix} u_2^{(k+1)} \\ p_2^{(k+1)} \end{pmatrix} &= b_{|\Omega_2} && \text{in } \Omega_2 \\ p_2^{(k+1)} &= p_1^{(k)} && \text{on } \Gamma_2 \\ p_2^{(k+1)} &= 0 && \text{on } \partial\Omega_2 \setminus \Gamma_2. \end{aligned}$$

Multiplicative Version

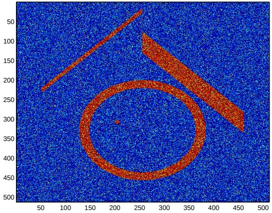
- using m colors, the number of divisions would be m times higher than the number of CPUs
- alternatively coloring similar to a checkerboard
- overlapping of domains of same color (but only 1 pixel)



Results



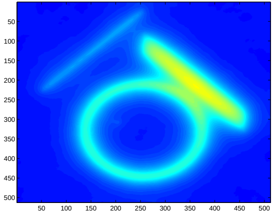
(a) Original



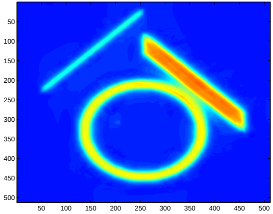
(b) Noisy

- using the following parameters:
  - $\lambda = 2$
  - $\epsilon = 10^{-2}$

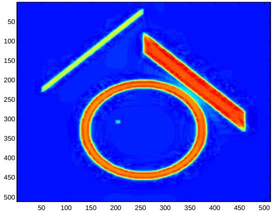
Results: Sequential Algorithm



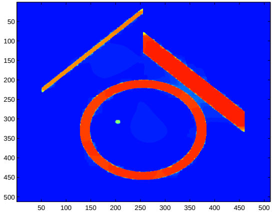
(a) After 1 iteration



(b) After 2 iterations

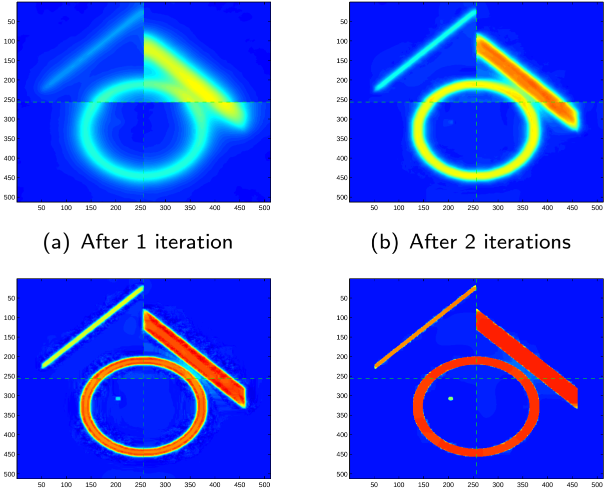


(c) After 3 iterations



(d) After 13 iterations

Results: Multiplicative version on 2 CPUs (4 domains)



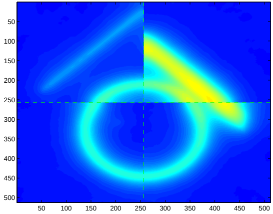
(a) After 1 iteration

(b) After 2 iterations

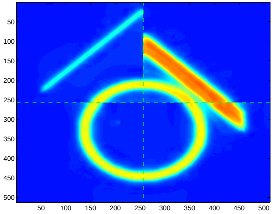
(c) After 3 iterations

(d) After 13 iterations

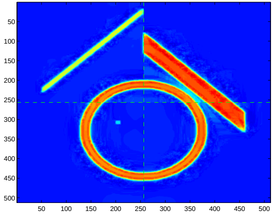
Results: Additive Version on 4 CPUs



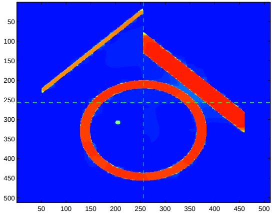
(a) After 1 iteration



(b) After 2 iterations

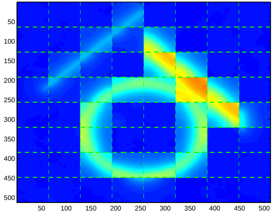


(c) After 3 iterations

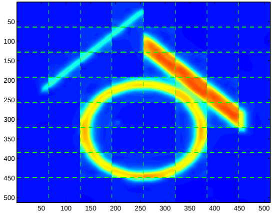


(d) After 13 iterations

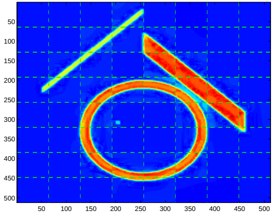
Results: Multiplicative Version on 32 CPUs (64 domains)



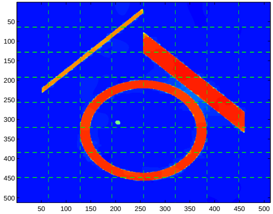
(a) After 1 iteration



(b) After 2 iterations

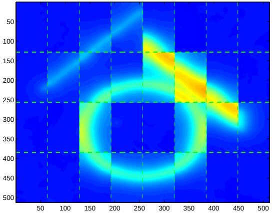


(c) After 3 iterations

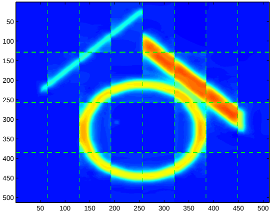


(d) After 14 iterations

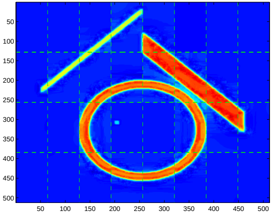
Results: Additive Version on 32 CPUs



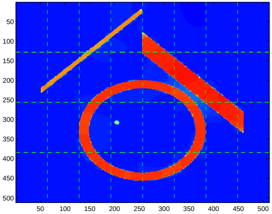
(a) After 1 iteration



(b) After 2 iterations

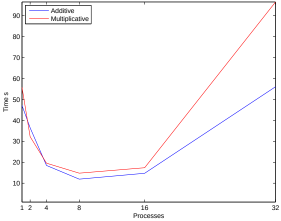


(c) After 3 iterations

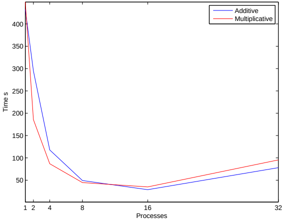


(d) After 16 iterations

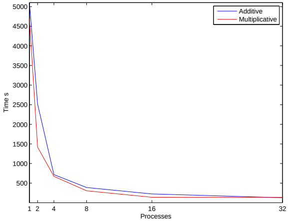
Computation Time



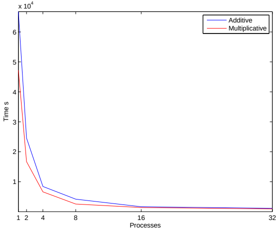
(a) Image size 512 × 512



(b) Image size 1024 × 1024

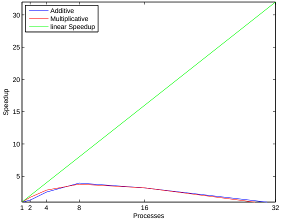


(c) Image size 2048 × 2048

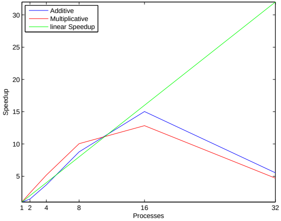


(d) Image size 4096 × 4096

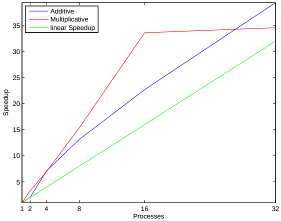
Speedup



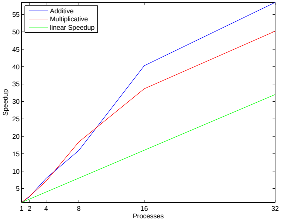
(a) Image size 512 × 512



(b) Image size 1024 × 1024



(c) Image size 2048 × 2048



(d) Image size 4096 × 4096