Lecture 6:

Nonlinear Isotropic Diffusion Filtering III: Efficient Sequential and Parallel Algorithms

Contents

Limitations of Explicit and Semi-Implicit Schemes

Explicit Scheme:

$$\frac{\mathbf{u}^{k+1} - \mathbf{u}^k}{\tau} = A(\mathbf{u}^k)\,\mathbf{u}^k.$$

- each step very cheap
- severe time step size restiction: $\tau < \frac{1}{\max\limits_{i \in J} |a_{ii}(\mathbf{u}^k)|}$.
- total efficiency: poor

Semi-Implicit Scheme:

$$\frac{\mathbf{u}^{k+1} - \mathbf{u}^k}{\tau} = A(u^k)\,\mathbf{u}^{k+1}$$

- satisfies discrete criteria (D1)–(D6) for all time step sizes (of course, accuracy decreases for larger time steps)
- requires to solve linear system $(I - \tau A(\mathbf{u}^k))\,\mathbf{u}^{k+1} = \mathbf{u}^k$
- total efficiency depends on costs for one step

One-Dimensional Case: The Thomas Algorithm

How Expensive is a One-Dimensional Semi-Implicit Step?

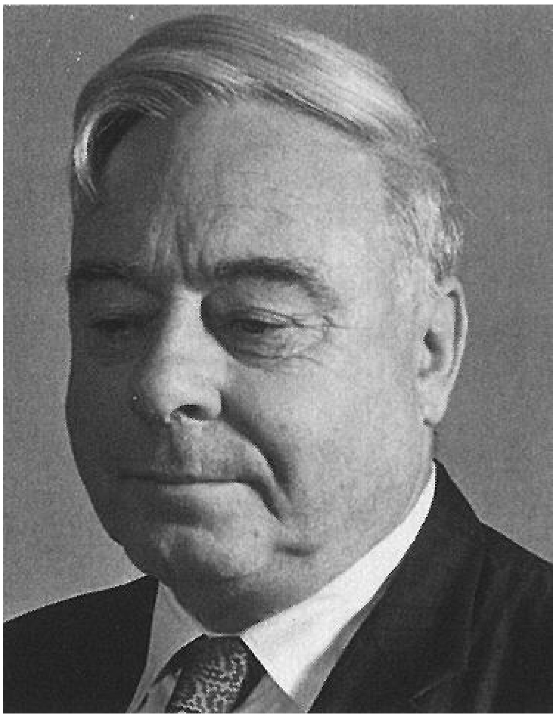
- $(I - \tau A(\mathbf{u}^k))\,\mathbf{u}^{k+1} \;=\; \mathbf{u}^k$ is a linear system where the system matrix $B := I - \tau A(\mathbf{u}^k)$ has a tridiagonal structure of nonvanishing entries:

$$B = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \gamma_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{N-2} & \alpha_{N-1} & \beta_{N-1} \\ & & & \gamma_{N-1} & \alpha_N \end{pmatrix}$$

  (Can you derive the detailed structure of the entries α_i, β_i, and γ_i ?)

- Already a simple Gaussian algorithm is highly efficient for solving tridiagonal systems of equations.
  In the English literature it is often called *Thomas algorithm*, since it has been used by L. H. Thomas in 1949.

M I A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27



Left: Carl–Friedrich Gauß (1777–1855) is regarded as one of the most famous mathematicians. Source: `www-gap.dcs.st-and.ac.uk/~history/PictDisplay/Gauss.html`. **Right:** Llewellen Hilleth Thomas (1903–1992) was a physicist, applied mathematician and computer scientist. He invented core memory in 1946. Source: `www.columbia.edu/acis/history/thomas.html`.

M I A 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

The Thomas Algorithm in Three Steps

We want to solve a tridiagonal linear system $B\mathbf{u} = \mathbf{c}$ with

$$B = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \gamma_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{N-2} & \alpha_{N-1} & \beta_{N-1} \\ & & & \gamma_{N-1} & \alpha_N \end{pmatrix}$$

This is done in three steps.

1. Decompose B in a lower bidiagonal matrix L and an upper bidiagonal matrix R: $B = LR$
2. Solve $L\mathbf{y} = \mathbf{c}$ for $\mathbf{y}$.
3. Solve $R\mathbf{u} = \mathbf{y}$ for $\mathbf{u}$.

M I A 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

Step 1: LR Decomposition

We decompose B in a product of a lower bidiagonal matrix L and an upper bidiagonal matrix R:

$$B = LR = \begin{pmatrix} 1 & & & & \\ l_1 & 1 & & & \\ & \ddots & \ddots & & \\ & & l_{N-2} & 1 & \\ & & & l_{N-1} & 1 \end{pmatrix} \begin{pmatrix} m_1 & r_1 & & & \\ & m_2 & r_2 & & \\ & & \ddots & \ddots & \\ & & & m_{N-1} & r_{N-1} \\ & & & & m_N \end{pmatrix}$$

$$= \begin{pmatrix} m_1 & r_1 & & & \\ l_1 m_1 & l_1 r_1 + m_2 & r_2 & & \\ & \ddots & \ddots & \ddots & \\ & & l_{N-2} m_{N-2} & l_{N-2} r_{N-2} + m_{N-1} & r_{N-1} \\ & & & l_{N-1} m_{N-1} & l_{N-1} r_{N-1} + m_N \end{pmatrix}$$

Semi-Implicit Scheme in One Dimension (5)

Comparing this with the coefficients of B shows that $r_i = \beta_i$ for $i = 1, \ldots, N-1$.

Moreover, the coefficients m_i and l_i can be computed via

$$
\begin{array}{l}
m_1 := \alpha_1 \\
\texttt{for} \;\; i = 1, 2, \ldots, N-1: \\
\qquad l_i := \gamma_i / m_i \\
\qquad m_{i+1} := \alpha_{i+1} - l_i \beta_i
\end{array}
$$

Semi-Implicit Scheme in One Dimension (6)

Step 2: Forward Elimination

We solve $L\mathbf{y} = \mathbf{c}$ for $\mathbf{y}$:

$$
\begin{pmatrix}
1 & & & & \\
l_1 & 1 & & & \\
 & \ddots & \ddots & & \\
 & & l_{N-2} & 1 & \\
 & & & l_{N-1} & 1
\end{pmatrix}
\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-1} \\ y_N \end{pmatrix}
=
\begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{N-1} \\ c_N \end{pmatrix}
$$

Proceeding from the first to the last equation gives

$$
\begin{array}{l}
y_1 := c_1 \\
\texttt{for} \;\; i = 2, 3, \ldots, N: \\
\qquad y_i := c_i - l_{i-1} y_{i-1}
\end{array}
$$

Step 3: Backward Substitution

We solve $R\mathbf{u} = \mathbf{y}$ for $\mathbf{u}$:

$$\begin{pmatrix} m_1 & r_1 & & & \\ & m_2 & r_2 & & \\ & & \ddots & \ddots & \\ & & & m_{N-1} & r_{N-1} \\ & & & & m_N \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-1} \\ y_N \end{pmatrix}$$

Proceeding from the last to the first equation gives

$$\begin{aligned} & u_N := y_N / m_N \\ & \texttt{for } i = N-1, N-2, ..., 1: \\ & \qquad u_i := (y_i - \beta_i u_{i+1}) / m_i \end{aligned}$$

where $r_i = \beta_i$ has been used.

All three steps together require just 10 lines of code!

MIA
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

Properties of the Thomas Algorithm

- The algorithm is stable for strictly diagonally dominant system matrices. This is the case for the 1-D semi-implicit discretisation.
- It can be regarded as a *recursive filter*:
  - step 1: determination of filter coefficients
  - step 2: causal filter $\longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow$
  - step 3: anticausal (acausal) filter $\longleftarrow \bullet \longleftarrow \bullet \longleftarrow \bullet \longleftarrow$
- The algorithm is highly efficient: It requires only
  - $(N-1) + 1 + (N-1) \;=\; 2N-1$ divisions
  - $(N-1) + (N-1) + (N-1) \;=\; 3N-3$ multiplications
  - $(N-1) + (N-1) + (N-1) \;=\; 3N-3$ subtractions

  Thus, its computational complexity is *linear* in N.
- The memory requirement is also linear in N.

MIA
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

Semi-Implicit Scheme in Higher Dimensions

- $A(u)$ remains sparse: In standard pixel ordering one obtains
  - a pentadiagonal system in 2-D (cf. Assignment T2, Problem 3),
  - a heptadiagonal system in 3-D.

  However, the bandwidth becomes large:
  many zero entries between the nonvanishing entries.

- direct algorithms fill in zeros within band
  $\Longrightarrow$ prohibitive storage and computational effort

- classical iterative algorithms (Jacobi, Gauß–Seidel, SOR):
  - convergence guaranteed
  - no additional storage effort
  - slow down for large τ (condition number increases)

- promising alternatives:
  - multigrid methods (implementation nontrivial)
  - splitting-based techniques

Additive Operator Splitting (AOS) Schemes

- Idea: decompose an m-D problem into simpler 1-D ones

- Let the m-dimensional expression

$$\operatorname{div}\left(g(|\nabla u_\sigma|^2)\,\nabla u\right) \;=\; \sum_{l=1}^{m} \partial_{x_l}\left(g(|\nabla u_\sigma|^2)\,\partial_{x_l} u\right)$$

  be discretised by the matrix–vector multiplication $A(\mathbf{u})\,\mathbf{u} = \sum_{l=1}^{m} A_l(\mathbf{u})\,\mathbf{u}$.

- AOS schemes consider instead of the semi-implicit scheme

$$\mathbf{u}^{k+1} \;=\; \left(I - \tau \sum_{l=1}^{m} A_l(\mathbf{u}^k)\right)^{-1} \mathbf{u}^k$$

  a split variant

$$\mathbf{u}^{k+1} \;=\; \frac{1}{m} \sum_{l=1}^{m} \left(I - m\tau A_l(\mathbf{u}^k)\right)^{-1} \mathbf{u}^k.$$

Additive Operator Splitting (AOS) Schemes (2)

Properties of AOS Schemes

- **Consistency**

  AOS scheme and semi-implicit one are not identical;
  but: same first order Taylor expansion in τ
  $\Longrightarrow$ consistency order $O(\tau + h_1^2 + \cdots + h_m^2)$.

- **Equal Treatment of All Axes**

  In contrast to multiplicative splittings such as

$$\mathbf{u}^{k+1} \;=\; \prod_{l=1}^{m} \Big(I - \tau A_l(\mathbf{u}^k)\Big)^{-1} \mathbf{u}^k.$$

- **Efficiency**

  AOS schemes compute

  - in each direction semi-implicit 1-D diffusions with step size $m\tau$ instead of τ.
  - final result: average

  Thomas algorithm applicable $\Longrightarrow$ linear complexity.
  About twice the effort of explicit scheme.

Additive Operator Splitting (AOS) Schemes (3)

- **Discrete Scale-Space Properties**

  Satisfy discrete requirements (D1)–(D6) from Lecture 5 for *all (!)* $\tau > 0$.
  Thus, the following holds:

  - Average Grey Level Invariance:

$$\frac{1}{N}\sum_j u_j^k \;=\; \frac{1}{N}\sum_j f_j \;=:\; \mu \qquad \forall\, k \in \mathbb{N}_0.$$

  - Maximum–Minimum Principle:

$$\min_j f_j \;\le\; u_i^k \;\le\; \max_j f_j \qquad \forall\, i,\ \forall\, k > 0$$

  - Lyapunov Sequences:

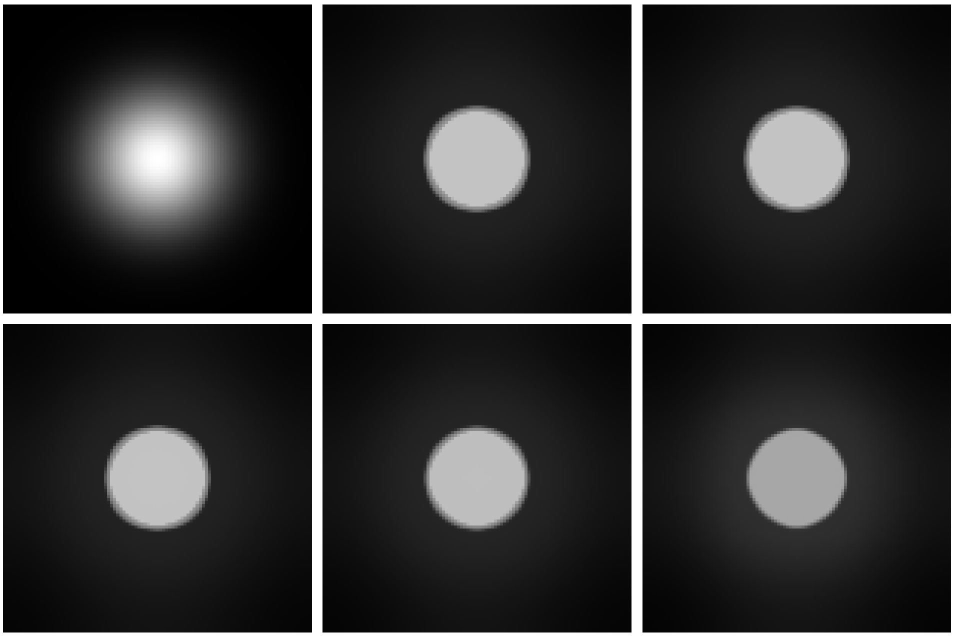
$$V^k \;:=\; \sum_i r(u_i^k)$$

    is decreasing and bounded from below for all convex $r \in \mathrm{C}(\mathbb{R})$.

  - Constant Steady-State:

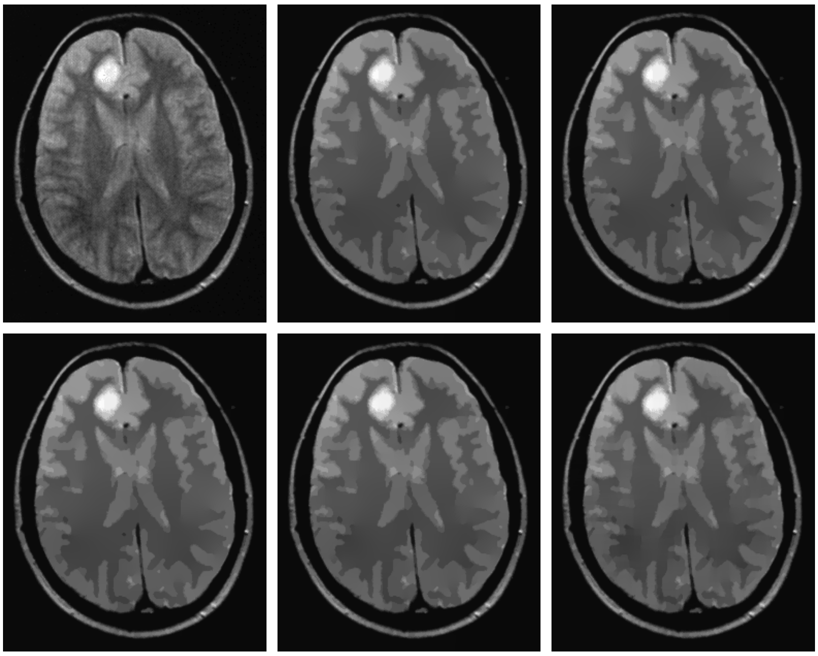
$$\lim_{k\to\infty} u_i^k = \mu \qquad \forall\, i.$$

Additive Operator Splitting (AOS) Schemes (4)



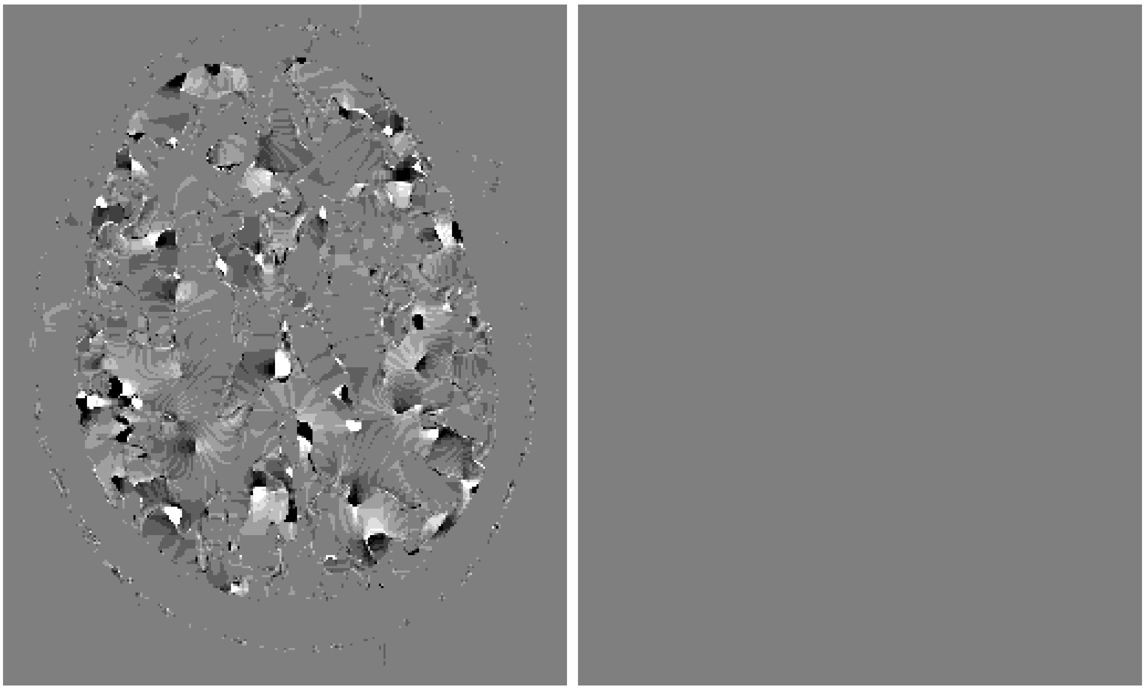
Isotropic nonlinear diffusion filtering of a Gaussian-like test image allows to evaluate the rotation invariance of AOS schemes ($\lambda = 8$, $\sigma = 1.5$). (a) **Top left:** Original image, $\Omega = (0, 101)^2$. (b) **Top middle:** Explicit scheme, 800 iterations, $\tau = 0.25$. (c) **Top right:** AOS scheme, 800 iterations, $\tau = 0.25$. (d) **Bottom left:** AOS scheme, 200 iterations, $\tau = 1$. (e) **Bottom middle:** AOS scheme, 40 iterations, $\tau = 5$. (f) **Bottom right:** AOS scheme, 10 iterations, $\tau = 20$.

Additive Operator Splitting (AOS) Schemes (5)



Isotropic nonlinear diffusion filtering of a medical image allows to evaluate the accuracy of AOS schemes ($\lambda = 2$, $\sigma = 1$). (a) **Top left:** Original image, $\Omega = (0, 255) \times (0, 308)$. (b) **Top right:** Explicit scheme, 800 iterations, $\tau = 0.25$. (c) **Middle left:** AOS scheme, 800 iterations, $\tau = 0.25$. (d) **Middle right:** AOS scheme, 200 iterations, $\tau = 1$. (e) **Bottom left:** AOS scheme, 40 iterations, $\tau = 5$. (f) **Bottom right:** AOS scheme, 10 iterations, $\tau = 20$.

Additive Operator Splitting (AOS) Schemes (6)



Multiplicative versus additive splittings. The difference between filtering prior to rotation by 90 degrees, and rotation prior to filtering is depicted ($\lambda = 2$, $\sigma = 1$, $\tau = 20$, 10 iterations). (a) **Left:** A multiplicative splitting such as LOD treats x and y axes differently. (b) **Right:** Additive operator splitting (AOS) treats both axes equally.

AOS Schemes on Parallel Computers (1)

AOS Schemes on Parallel Computers

Coarse Grain Parallelism

- AOS scheme of an m-dimensional diffusion process is given by

$$\mathbf{u}^{k+1} \;=\; \frac{1}{m}\sum_{l=1}^{m}\left(I - m\tau A_l(\mathbf{u}^k)\right)^{-1}\mathbf{u}^k.$$

- It averages m results of type

$$\mathbf{v}_l^{k+1} := \left(I - m\tau A_l(\mathbf{u}^k)\right)^{-1}\mathbf{u}^k \qquad (l = 1, ..., m).$$

- Computing these $\mathbf{v}_l^{k+1}$ is possible on different processors.

M I A 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

AOS Schemes on Parallel Computers (2)

Mid Grain Parallelism

- decomposition of $(I - m\tau A_l(\mathbf{u}^k))^{-1}$:
  $\prod_{j \neq l} N_j$ one-dimensional diffusions along l direction (N_j: pixels in j direction).
- processes are completely independent
- all tridiagonal systems have equal length $\Longrightarrow$ perfect work balance

AOS Schemes on Parallel Computers (3)

Example on a Distributed Memory Architecture (2002)

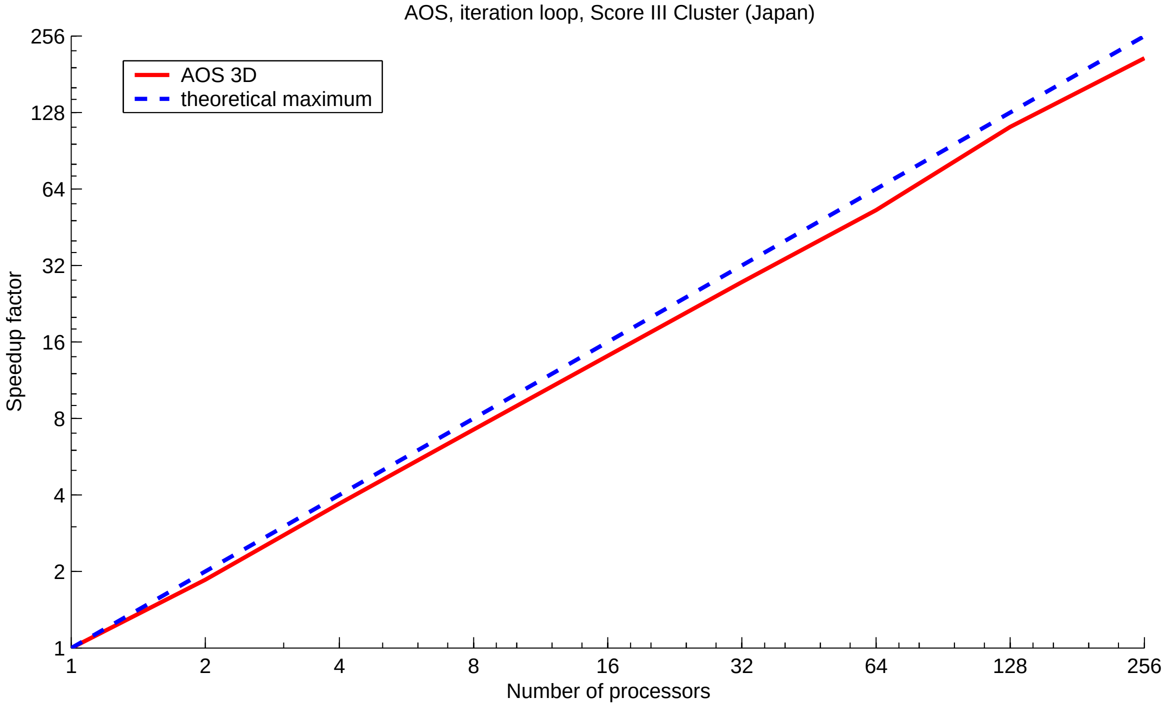
- Score III Cluster (Tsukuba, Japan):
  - low latency network (Myrinet 2000) with 524 nodes (Pentium 3, 933 MHz)
  - rank 40 in the 2002 Top 500 list of supercomputers
- 3-D AOS scheme for data cube of size $256 \times 256 \times 128$
- speed-up factor of 209 for 256 processors.
- communication volumes of up to 1.83 GByte per second

Runtimes for 3-D AOS, 10 iterations

Processors	1	2	4	8	16	32	64	128	256
Runtime [s]	212.741	114.625	57.534	29.401	15.065	7.731	4.029	1.894	1.017

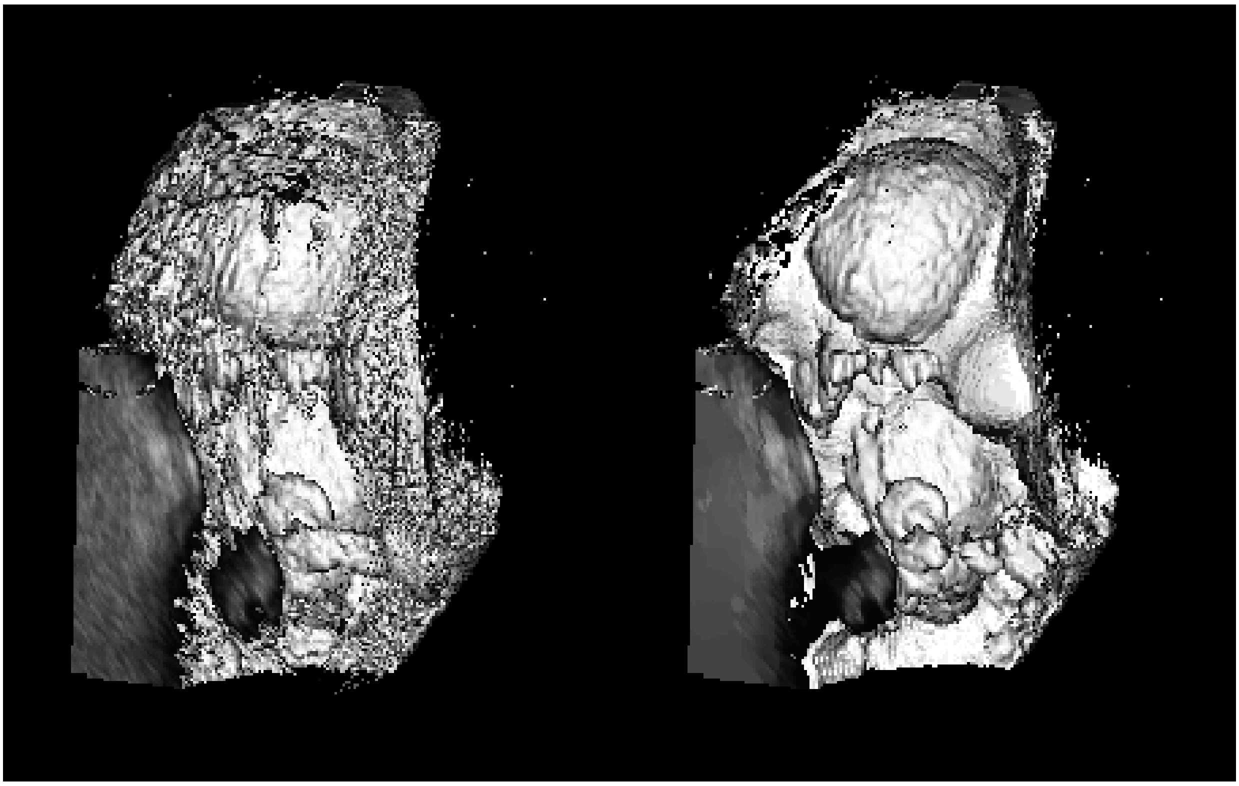
M I A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

AOS Schemes on Parallel Computers (4)



Speedup chart.

AOS Schemes on Parallel Computers (5)



Example of a result from a parallel AOS scheme in 3-D. Rendering of a 3-D ultrasound image of a 10-weeks old fetus. **Left:** Original. **Right:** Filtered.

M I A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

Summary

- AOS schemes decompose an m-D problem into a sum of simpler 1-D ones.
- semi-implicit 1-D diffusion processes solved efficiently by Thomas algorithm
- can be regarded as recursive filtering
- equal treatment of all axes
- good rotational invariance
- create discrete diffusion scale-spaces for all time step sizes τ
- one order of magnitude more efficient than explicit schemes

scheme	formula	stability	costs/iter.	efficiency
explicit	$\mathbf{u}^{k+1} = (I+\tau \sum_{l=1}^{m} A_l(\mathbf{u}^k))\,\mathbf{u}^k$	$\tau < \frac{h^2}{2m}$	very low	low
semi-implicit	$\mathbf{u}^{k+1} = (I-\tau \sum_{l=1}^{m} A_l(\mathbf{u}^k))^{-1}\mathbf{u}^k$	$\tau < \infty$	medium	fair
AOS	$\mathbf{u}^{k+1} = \frac{1}{m} \sum_{l=1}^{m}(I-m\tau A_l(\mathbf{u}^k))^{-1}\mathbf{u}^k$	$\tau < \infty$	low	high

- coarse and mid grain parallelism

References

- L. H. Thomas: *Elliptic problems in linear difference equations over a network*. Technical Report, Watson Scientific Computing Laboratory, Columbia University, New York, NJ, 1949.
  (original paper by Thomas)
- H. R. Schwarz: *Numerische Mathematik*. Vierte Auflage, Teubner, Stuttgart, 1997.
  (detailed description of the Thomas algorithm)
- T. Lu, P. Neittaanmäki, X.-C. Tai: A parallel splitting up method and its application to Navier–Stokes equations. *Applied Mathematics Letters*, Vol. 4, 25-29, 1991.
  (early paper on AOS ideas for fluid mechanical problems)
- J. Weickert, B. M. ter Haar Romeny, M. A. Viergever: Efficient and reliable schemes for nonlinear diffusion filtering, *IEEE Transactions on Image Processing*, Vol. 7, 398-410, 1998.
  (`www.mia.uni-saarland.de/weickert/publications.html`)
  (describes the foundations and algorithmic details of AOS schemes on sequential computers)
- A. Bruhn, T. Jacob, M. Fischer, T. Kohlberger, J. Weickert, U. Brüning, C. Schnörr: Designing 3-D nonlinear diffusion filters for high performance cluster computing. L. Van Gool (Ed.), *Pattern Recognition*, Lecture Notes in Computer Science, Vol. 2449, Springer, Berlin, 290-297, 2002.
  (`www.mia.uni-saarland.de/weickert/publications.html`)
  (AOS schemes on a Myrinet cluster with 256 PCs)

M I A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

Assignment T2 – Theoretical Home Work

Problem 1 (Isotropic Nonlinear Diffusion: Diffusivities) (3 points)

Analyse the given diffusivities $g(s^2)$ with respect to the possibility of forward-backward diffusion. To this end consider the sign of $\Phi'(s^2)$ for the flux function $\Phi(s^2) = s\, g(s^2)$. For which values of u_x, local contrast enhancement is possible for

(a) the rational Perona-Malik diffusivity $g(s^2) := \frac{1}{1+s^2/\lambda^2}$,

(b) the Charbonnier diffusivity $g(s^2) := \frac{1}{\sqrt{1+s^2/\lambda^2}}$,

(c) an exponential Perona-Malik diffusivity $g(s^2) := \exp\left(\frac{-s^2}{2\lambda^2}\right)$?

For an additional visualisation of the behaviour of $\Phi(s^2)$, you can sketch the graphs of these functions or plot them with `gnuplot`.

Problem 2 (Isotropic Nonlinear Diffusion: Stencil Notation of Explicit Scheme) (2 points)

Write down the explicit scheme for 2-D isotropic nonlinear diffusion filtering in stencil notation for some inner pixel and some arbitrary diffusivity g.

Problem 3 (Isotropic Nonlinear Diffusion: Matrix Notation of Semi-Implicit Scheme) (3 points)

Semi-implicit nonlinear diffusion filtering creates a linear system of equations. Write down the system matrix for a 1-D signal of size 4 and a 2-D image of size 3×4. You can assume Neumann boundary conditions with reflected dummy pixels. In 2-D, use a consecutive pixel numbering such as $k(i,j) \;:=\; (i-1) + 3(j-1) + 1$.

Problem 4 (Problem 4 (Linear Systems of Equations and Thomas' Algorithm) (4 points)

In this problem you should derive the analogon of the Thomas' algorithm for solving a linear system of equations with a pentadiagonal 5×5 system matrix. To this end proceed in the following two steps:

(a) Decompose the system matrix

$$A := \begin{pmatrix} \gamma_1 & \delta_1 & \varepsilon_1 & 0 & 0 \\ \beta_1 & \gamma_2 & \delta_2 & \varepsilon_2 & 0 \\ \alpha_1 & \beta_2 & \gamma_3 & \delta_3 & \varepsilon_3 \\ 0 & \alpha_2 & \beta_3 & \gamma_4 & \delta_4 \\ 0 & 0 & \alpha_3 & \beta_4 & \gamma_5 \end{pmatrix}$$

in the upper and lower tridiagonal matrices

$$L := \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ b_1 & 1 & 0 & 0 & 0 \\ a_1 & b_2 & 1 & 0 & 0 \\ 0 & a_2 & b_3 & 1 & 0 \\ 0 & 0 & a_3 & b_4 & 1 \end{pmatrix} \quad \text{and} \quad R := \begin{pmatrix} c_1 & d_1 & e_1 & 0 & 0 \\ 0 & c_2 & d_2 & e_2 & 0 \\ 0 & 0 & c_3 & d_3 & e_3 \\ 0 & 0 & 0 & c_4 & d_4 \\ 0 & 0 & 0 & 0 & c_5 \end{pmatrix}$$

and derive the formulae for calculating the entries of L and R. Pay attention to the ordering of calculations: Only use values which are already known in each step.

(b) In a second step, use this to solve a linear system $A\mathbf{x} = \mathbf{f}$. How does the forward and backward substitution look like in this case?

Deadline for submission: Friday, May 9, 10 am (before the lecture).