# Under pressure!

Fast pressure calculations in the Deft package

Jos van Kan

j.vankan@math.tudelft.nl

Delft Institute of Applied mathematics (DIAM)

TU Delft

# Deft package

Delft Flow and Transport

- Navier Stokes equations for incompressible flow on general domains
- Offshoot of ISNaS (Information System Navier Stokes)

# Design decisions

- Finite Volume Method
- Rectangular blocks of curvilinear coordinates
- Staggered grid
- Time-dependent algorithm
- Pressure correction for the incompressibility condition

# Navier Stokes equations

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}^T) + \nabla p = \nabla \cdot T + \mathbf{f}$$
$$\nabla \cdot \mathbf{u} = 0$$

- $\mathbf{u}$ velocities
- $p$ pressure
- $T$ stress tensor
- $\mathbf{f}$ body forces like gravity

Variations in density are not taken into account. $p$ and $T$ are scaled quantities.

# Method of lines

$$\frac{d\mathbf{u}_h}{dt} + G\mathbf{p}_h = N(\mathbf{u}_h) + L(\mathbf{u}_h) + \mathbf{f}_h$$
$$D\mathbf{u}_h = 0$$

# Time Integration

Implicit time integration, like for instance Crank-Nicolson

$$\mathbf{u}_h^{n+1} + \Delta t G \mathbf{p}_h^{n+1/2} =$$

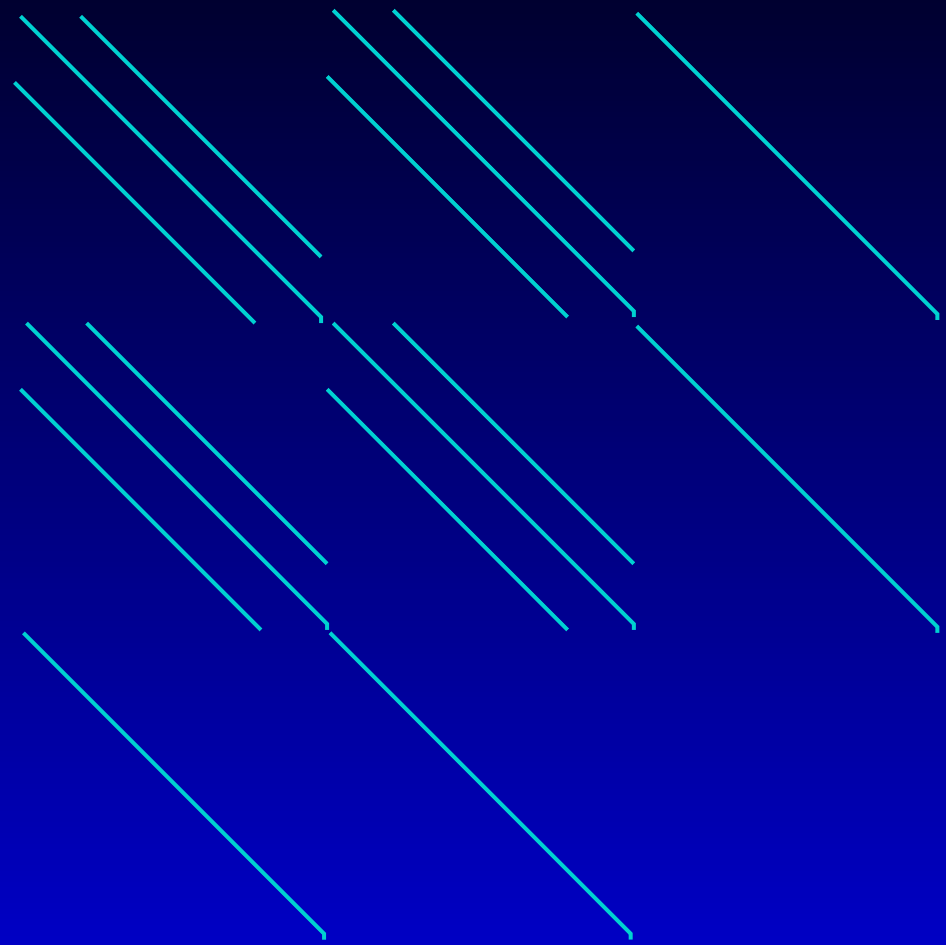
$$\mathbf{u}_h^{n} + \frac{1}{2}\Delta t \left( N(\mathbf{u}_h^{n+1}) + L(\mathbf{u}_h^{n+1}) + \mathbf{f}_h^{n+1} \right) +$$

$$\frac{1}{2}\Delta t \left( N(\mathbf{u}_h^{n}) + L(\mathbf{u}_h^{n}) + \mathbf{f}_h^{n} \right)$$

$$D\mathbf{u}_h^{n+1} = 0$$

TUDelft

# Matrix Structure



# Pressure Correction

$$\mathbf{u}_h^* + \Delta t G \mathbf{p}_h^{n-1/2} =$$

$$\mathbf{u}_h^n + \frac{1}{2}\Delta t \left( N(\mathbf{u}_h^*) + L(\mathbf{u}_h^*) + \mathbf{f}_h^{n+1} \right) +$$

$$\frac{1}{2}\Delta t \left( N(\mathbf{u}_h^n) + L(\mathbf{u}_h^n) + \mathbf{f}_h^n \right)$$

$$\mathbf{u}_h^{n+1} - \mathbf{u}_h^* + \Delta t G \Delta \mathbf{p} = 0$$

$$-D\mathbf{u}_h^* + \Delta t D G \Delta \mathbf{p} = 0$$

$$\mathbf{p}^{n+1/2} = \mathbf{p}^{n-1/2} + \Delta \mathbf{p}$$

TUDelft

# Pressure Correction

$$\mathbf{u}_h^* + \Delta t G \mathbf{p}_h^{n-1/2} =$$

$$\mathbf{u}_h^n + \frac{1}{2}\Delta t \left( N(\mathbf{u}_h^*) + L(\mathbf{u}_h^*) + \mathbf{f}_h^{n+1} \right) +$$

$$\frac{1}{2}\Delta t \left( N(\mathbf{u}_h^n) + L(\mathbf{u}_h^n) + \mathbf{f}_h^n \right)$$

$$\mathbf{u}_h^{n+1} - \mathbf{u}_h^* + \Delta t G \Delta \mathbf{p} = 0$$

$$-D\mathbf{u}_h^* + \Delta t DG \Delta \mathbf{p} = 0$$

$$\mathbf{p}^{n+1/2} = \mathbf{p}^{n-1/2} + \Delta \mathbf{p}$$

TUDelft

# $p$ Matrix structure 2D

Example: $n \times n$ block, $N = n^2$.

- Tridiagonal block matrix of tridiagonal matrices
- Bandwidth: $O(\sqrt{N})$
- Flops $LU$ decomposition: $O(N^2)$. (One time only on fixed domains)
- Flops $LU$ backsubstitution: $O(N^{3/2})$
- Flops matrix vector multiplication: $O(N)$.

TUDelft

# $p$ Matrix structure (3D)

Example: $n \times n \times n$ block, $N = n^3$.

- Tridiagonal block matrix of tridiagonal blockmatrices of tridiagonal matrices
- Bandwidth $O(N^{2/3})$
- Flops $LU$ decomposition: $O(N^{7/3})$. (Only once on fixed domains)
- Flops $LU$ backsubstitution $O(N^{5/3})$
- Flops matrix vector multiplication: $O(N)$

# A Classic: Defect Correction

Solve $A\mathbf{x} = \mathbf{b}$

Presets: $\mathbf{x}^0 = 0, \mathbf{r}^0 = \mathbf{b} - A\mathbf{x}^0 = \mathbf{b}, k = 0$

**while** $\|r^k\|_\infty > \varepsilon \|b\|_\infty$ **do**

Solve $P\mathbf{c}^k = \mathbf{r}^k$ {P is preconditioner}

$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{c}^k$

$\mathbf{r}^{k+1} = \mathbf{r}^k - A\mathbf{c}^k$

$k = k + 1$

**end while**

Each iteration requires $O(N)$ flops.

# Preconditioners

**Classic** With $A = D - L - U$

- Jacobi: $P = D$
- Gauss-Seidel: $P = D - L$
- Successive overrelaxation: $P = (D/\omega - L)$

**Modern** With $A = LU$

- Incomplete LU (ILU):
  $A = \tilde{L}\tilde{U} + E$, $P = \tilde{L}\tilde{U}$. $\tilde{L}$ and $\tilde{U}$ sparse, usually the same sparsity pattern as $A$.
- Incomplete Block LU (IBLU).

# DC Error Reduction

$$\mathbf{r}^{k+1} = \mathbf{r}^k - A\mathbf{c}^k = (I - AP^{-1})\mathbf{r}^k$$

$$\mathbf{A}^{-1}\mathbf{b} - \mathbf{x}^{k+1} = A^{-1}\mathbf{r}^{k+1} = \varepsilon^{k+1}$$

$$\varepsilon^{k+1} = A^{-1}(I - AP^{-1})A\varepsilon^k = (I - P^{-1}A)\varepsilon^k$$

Reduction governed by spectral radius of $(I - AP^{-1})$.
For the Laplacian:

- Jacobi and Gauss-Seidel: $1 - O(h^2)$
- SOR with optimal $\omega$ and a whole slew of other conditions: $1 - O(h)$.

.

TU Delft

# Effectiveness of DC

How many iterations to gain a decimal digit?

$$\varepsilon^{n+k} = \lambda_1^k \varepsilon^n$$

$$\lambda_1^k = 0.1$$

$$k \log \lambda_1 = -\log 10$$

$$k = -\frac{2.3}{\log \lambda_1}$$

$$k = O(\frac{1}{1-\lambda_1})$$

TU Delft

# Effectiveness of DC

Jacobi and Gauss-Seidel $O(h^{-2})$ iterations.
SOR $O(h^{-1})$ iterations.
In 2D:

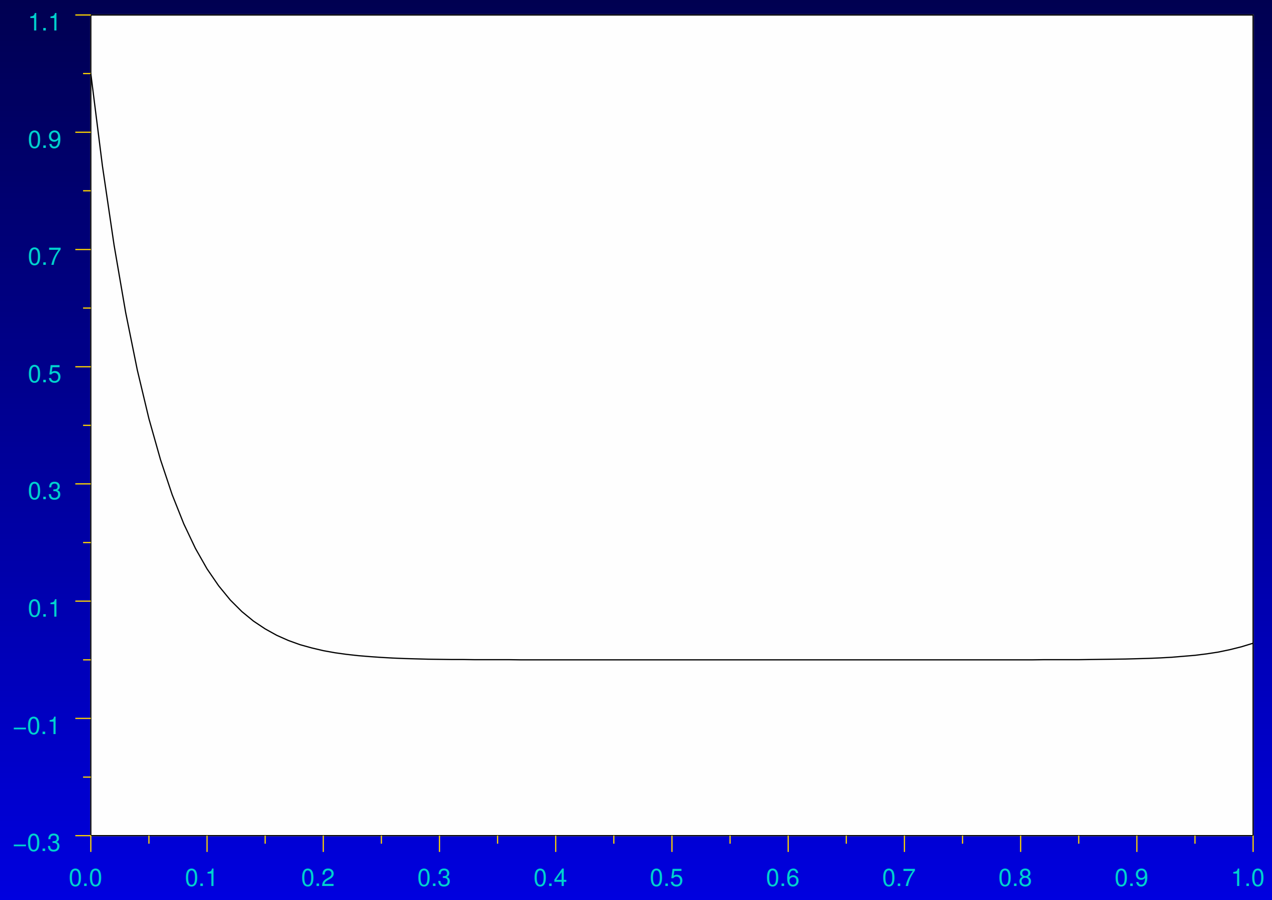
- Jacobi and GS $O(N^2)$ flops, worse than $LU$
- SOR $(O(N^{3/2})$ flops, order equal to $LU$

In 3D:

- Jacobi and GS $O(N^{5/3})$ flops, order equal to $LU$
- SOR $O(N^{4/3})$ flops, better than $LU$

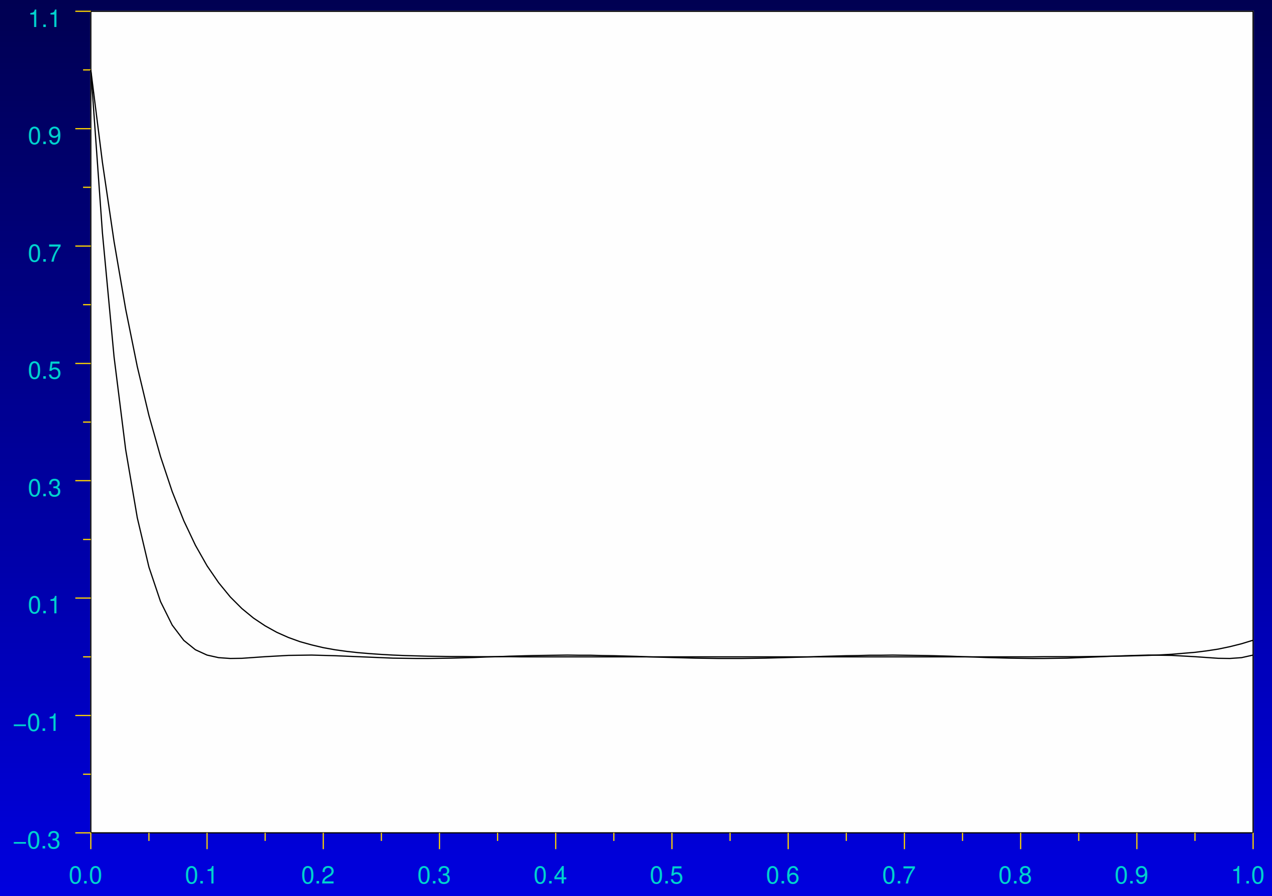
# Convergence properties

Damped Jacobi, 10 iterations



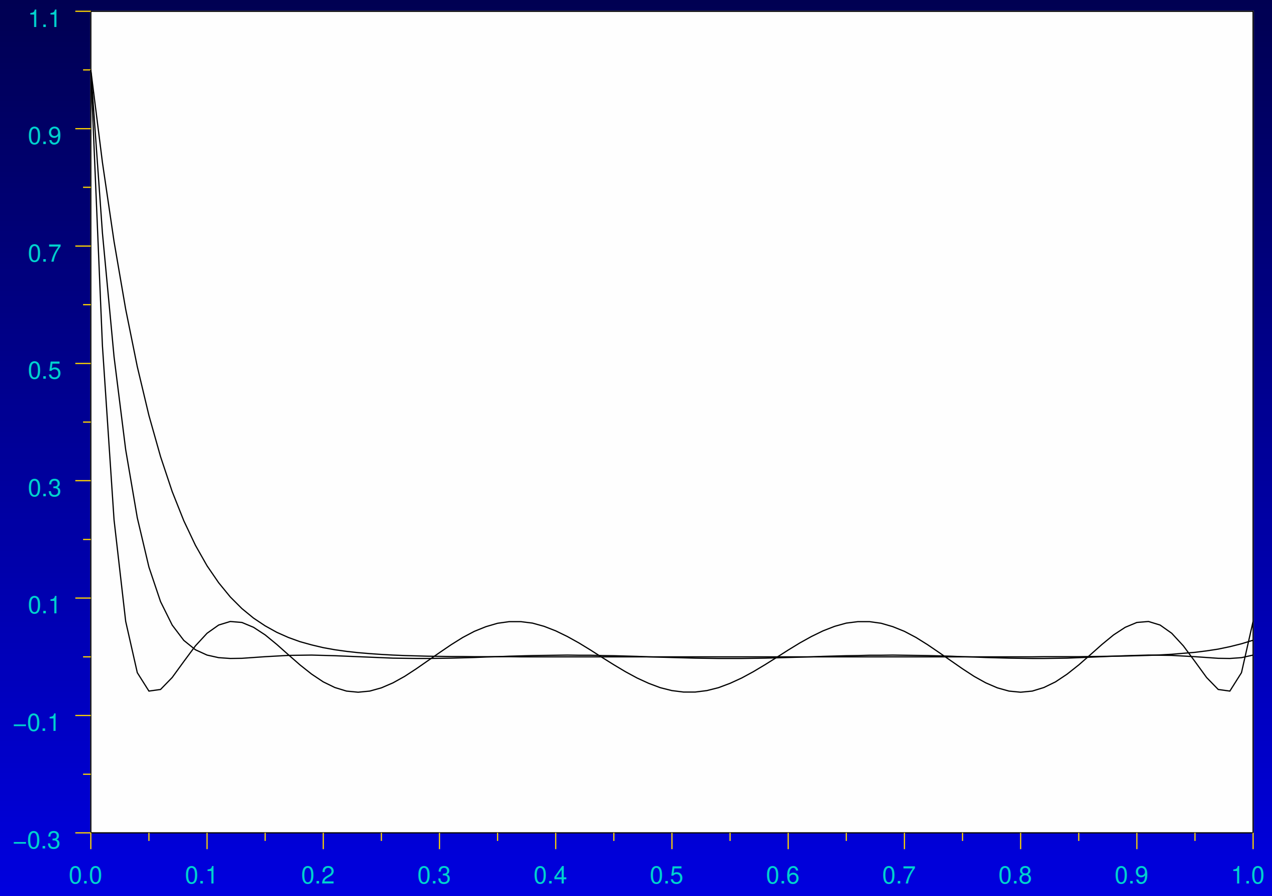
# Convergence properties

Chebyshev10, $\lambda_0 = 0.1$



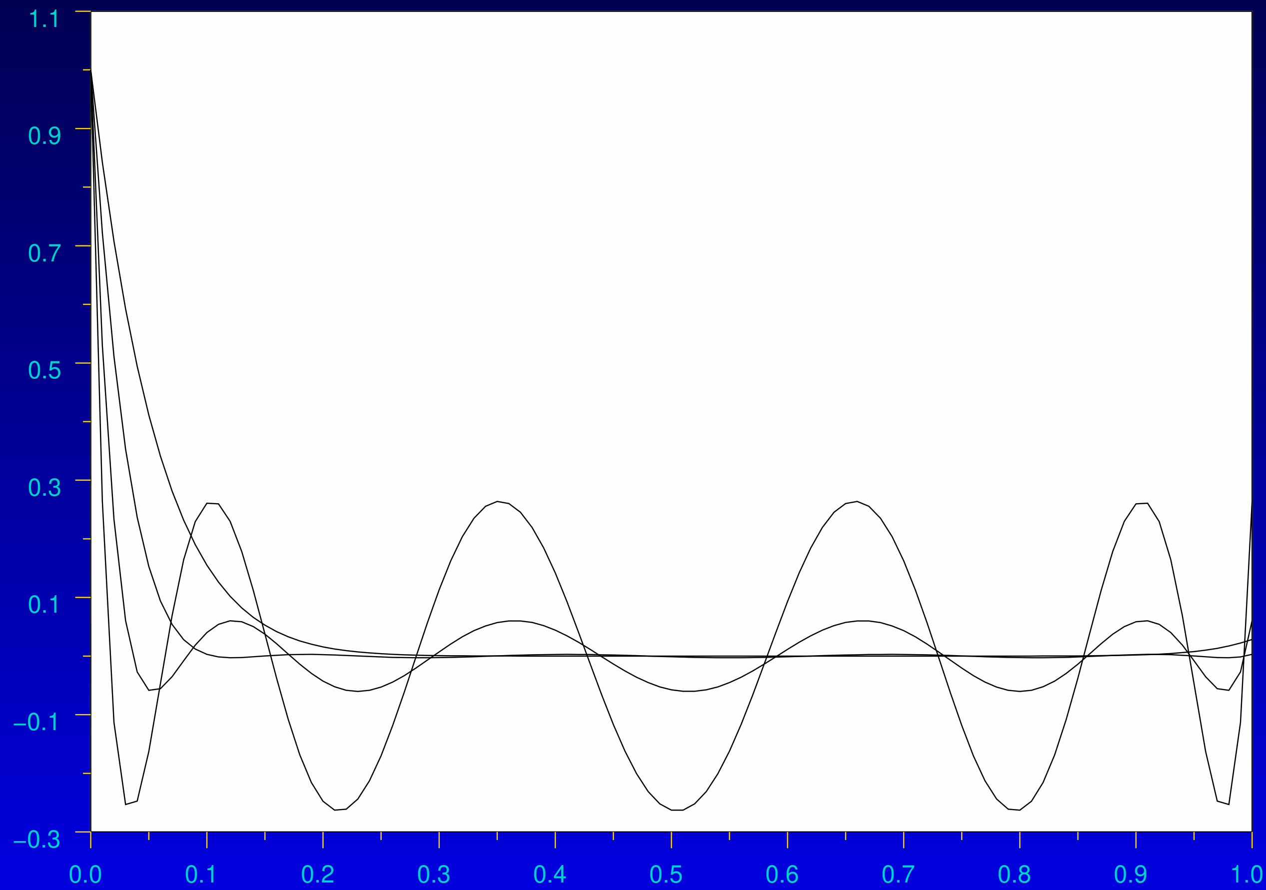
# Convergence properties

Chebyshev10, $\lambda_0 = 0.03$

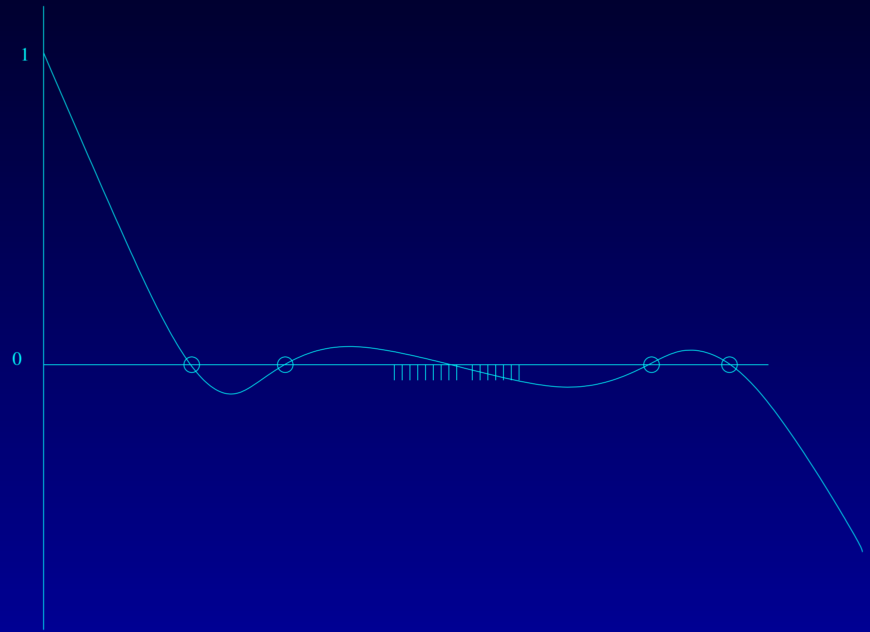


# Convergence properties

Chebyshev10, $\lambda_0 = 0.01$



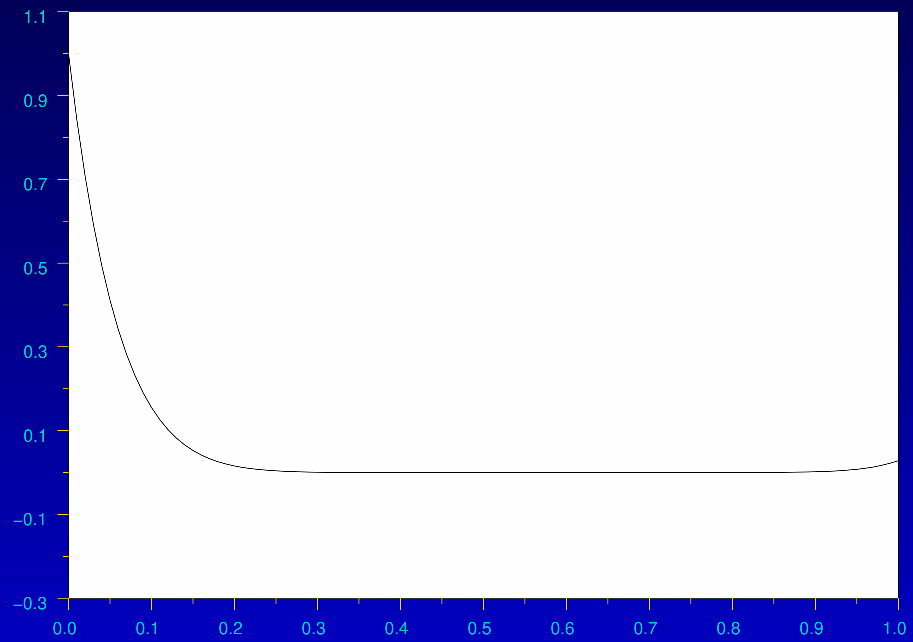
# Gradient Methods




- "Best" polynomial on the fine structure of the spectrum
- Gradient Methods are always better than Defect Correction
- Irregular convergence behaviour

# Multigrid

Defect Correction: very effective on a part of the spectrum.




The eigenspace of the spectral interval $(0.2, 1)$ is virtually reduced to 0 in a few iterations.

# 1D example

Consider $-\frac{d^2u}{dx^2} = f, \quad u(0) = u(1) = 0$
Discretize into $N$ intervals): $A\mathbf{u} = \mathbf{f}$
Eigenvalues of $1 - P^{-1}A$ are

$$\lambda_k = 1 - \sin^2 \frac{k\pi}{2N}, k = 1, \ldots, N-1.$$

Corresponding eigenvectors

$$v_{kj} = \sin \frac{kj\pi}{N}, k, j = 1, \ldots, N-1.$$

Eigenvalues close to 1 correspond to smooth eigenvectors, also for the Laplacian in 2 and 3D.

# Rough and smooth spectrum

- Rough part of the spectrum: defect correction, smoother in MG speak.
- Smooth part of the spectrum: solve problem on coarser grid and interpolate. Coarse grid correction in MG speak.

# Restriction and prolongation

Fine grid correction: $A_h \mathbf{c}_h = \mathbf{r}_h$
Coarse grid correction $A_H \mathbf{c}_H = \mathbf{r}_H$
Transfer operators:

- $P_{hH}$: prolongation from coarse to fine grid. Interpolation usually.
- $R_{Hh}$: restriction from fine grid to coarse grid. $R = P^T$ in symmetric problems.

The coarse grid correction becomes:
$R_{Hh} A_h P_{hH} \mathbf{c}_H = R_{Hh} \mathbf{r}_h$

# Two Grid Algorithm

Presets: $\mathbf{u}_h^0$, $\mathbf{r}_h^0 = \mathbf{f}_h - A\mathbf{u}_h^0$

$\mathbf{u}_h^{\text{prs}} = S(\mathbf{u}_h^0, \mathbf{b}, A, n_0)\{\text{Presmoothing}\}$

$\mathbf{r}_H = R_{Hh}\mathbf{r}_h$

# Two Grid Algorithm

Presets: $\mathbf{u}_h^0, \mathbf{r}_h^0 = \mathbf{f}_h - A\mathbf{u}_h^0$
$\mathbf{u}_h^{\text{prs}} = S(\mathbf{u}_h^0, \mathbf{b}, A, n_0)$ {Presmoothing}
$\mathbf{r}_H = R_{Hh}\mathbf{r}_h$
Solve $A_H\mathbf{c}_H = \mathbf{r}_H$
$\mathbf{u}_h^{\text{cgc}} = \mathbf{u}_h^{\text{prs}} + P_{hH}\mathbf{c}_H$ {Coarse Grid Correction}

# Two Grid Algorithm

Presets: $\mathbf{u}_h^0, \mathbf{r}_h^0 = \mathbf{f}_h - A\mathbf{u}_h^0$
$\mathbf{u}_h^{\text{prs}} = S(\mathbf{u}_h^0, \mathbf{b}, A, n_0)$ {Presmoothing}
$\mathbf{r}_H = R_{Hh}\mathbf{r}_h$
Solve $A_H\mathbf{c}_H = \mathbf{r}_H$
$\mathbf{u}_h^{\text{cgc}} = \mathbf{u}_h^{\text{prs}} + P_{hH}\mathbf{c}_H$ {Coarse Grid Correction}
$\mathbf{u}_h^{\text{pos}} = S(\mathbf{u}_h^{\text{cgc}}, \mathbf{b}, A, n_1)$ {Postsmoothing}

# Multi Grid Algorithm

**Require:** $A_{\ell+1} = R_{\ell+1,\ell} A_\ell P_{\ell,\ell+1}$ have been calculated on all levels

**MGRecursive** $(A_\ell, \mathbf{r}_\ell, \mathbf{c}_\ell, \ell)$

**if** $\ell < p$ **then**

**else**

Solve $A_p \mathbf{c}_p = \mathbf{r}_p$ {Direct solution on coarsest level}

**end if**

# Multi Grid Algorithm

**Require:** $A_{\ell+1} = R_{\ell+1,\ell} A_\ell P_{\ell,\ell+1}$ have been calculated on all levels

**MGRecursive** $(A_\ell, \mathbf{r}_\ell, \mathbf{c}_\ell, \ell)$

**if** $\ell < p$ **then**

$\mathbf{c}_\ell = S(\mathbf{0}, \mathbf{r}_\ell, A_\ell, n_0)$ {Presmoothing}

$\mathbf{r}_{\ell+1} = R_{\ell+1,\ell}(\mathbf{r}_\ell - A_\ell \mathbf{c}_\ell)$ {Calculate coarse grid residual}

**else**

Solve $A_p \mathbf{c}_p = \mathbf{r}_p$ {Direct solution on coarsest level}

**end if**

TUDelft

# Multi Grid Algorithm

**Require:** $A_{\ell+1} = R_{\ell+1,\ell} A_\ell P_{\ell,\ell+1}$ have been calculated on all levels

**MGRecursive** $(A_\ell, \mathbf{r}_\ell, \mathbf{c}_\ell, \ell)$

**if** $\ell < p$ **then**

$\mathbf{c}_\ell = S(\mathbf{0}, \mathbf{r}_\ell, A_\ell, n_0)$ {Presmoothing}

$\mathbf{r}_{\ell+1} = R_{\ell+1,\ell}(\mathbf{r}_\ell - A_\ell \mathbf{c}_\ell)$ {Calculate coarse grid residual}

call MGRecursive $(A_{\ell+1}, \mathbf{r}_{\ell+1}, \mathbf{c}_{\ell+1}, \ell + 1)$

$\mathbf{c}_\ell = \mathbf{c}_\ell + P_{\ell,\ell+1} \mathbf{c}_{\ell+1}$ {Coarse grid correction}

**else**

Solve $A_p \mathbf{c}_p = \mathbf{r}_p$ {Direct solution on coarsest level}

**end if**

# Multi Grid Algorithm

**Require:** $A_{\ell+1} = R_{\ell+1,\ell} A_\ell P_{\ell,\ell+1}$ have been calculated on all levels

**MGRecursive** $(A_\ell, \mathbf{r}_\ell, \mathbf{c}_\ell, \ell)$

**if** $\ell < p$ **then**

$\mathbf{c}_\ell = S(\mathbf{0}, \mathbf{r}_\ell, A_\ell, n_0)$ {Presmoothing}

$\mathbf{r}_{\ell+1} = R_{\ell+1,\ell}(\mathbf{r}_\ell - A_\ell \mathbf{c}_\ell)$ {Calculate coarse grid residual}

call MGRecursive $(A_{\ell+1}, \mathbf{r}_{\ell+1}, \mathbf{c}_{\ell+1}, \ell + 1)$

$\mathbf{c}_\ell = \mathbf{c}_\ell + P_{\ell,\ell+1} \mathbf{c}_{\ell+1}$ {Coarse grid correction}

$\mathbf{c}_\ell = S(\mathbf{c}_\ell, \mathbf{r}_\ell, A_\ell, n_1)$ {Postsmoothing}

**else**

Solve $A_p \mathbf{c}_p = \mathbf{r}_p$ {Direct solution on coarsest level}

**end if**

# The MG miracle

- Spectrum of $I - P^{-1}A$ bounded away from 1 uniformly in $h$.
- Number of iterations does not depend on $h$.
- The workload is theoretically $O(N)$ flops.

But how big is the multiplicative constant going to be?

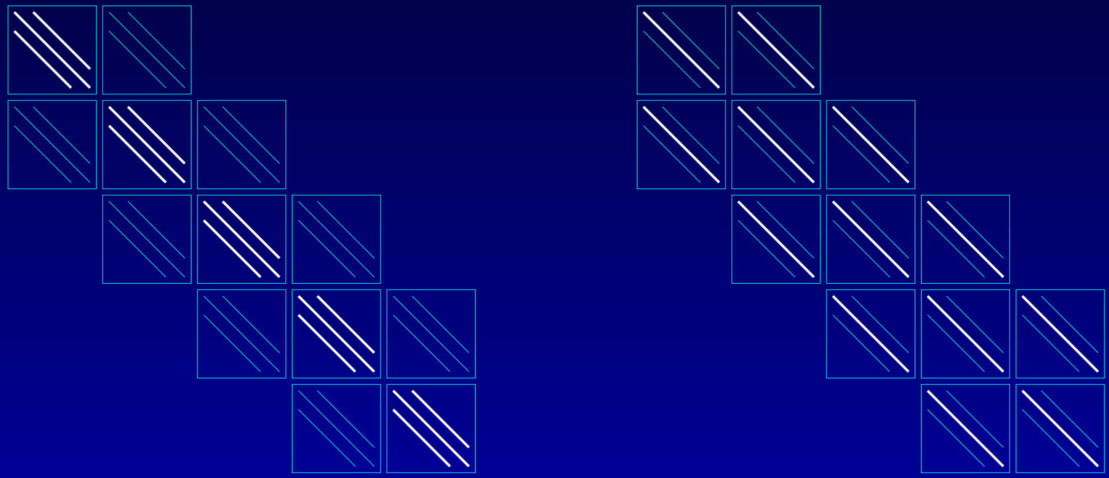
# Robust Blackbox

Wishlist:

- Good smoother under various circumstances (anisotropy, stretched and skew cells)
- Arbitrary number of points in either direction

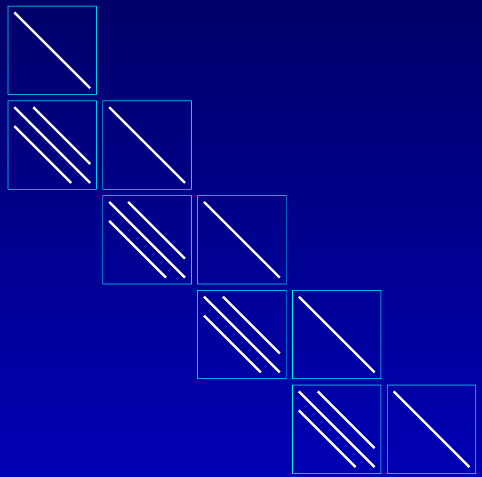
# Smoothers tested

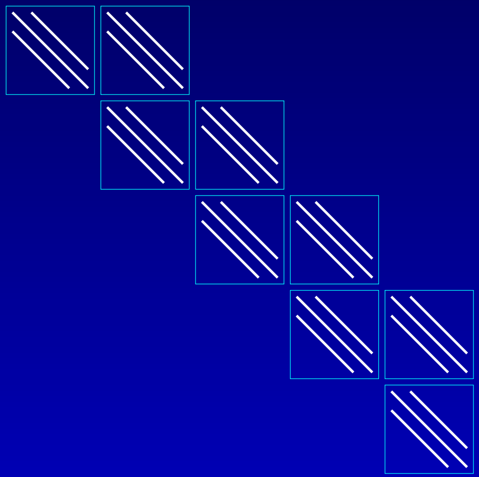
- (Alternating) damped line Jacobi, 1 or 2 postsmoothing steps, no presmoothing



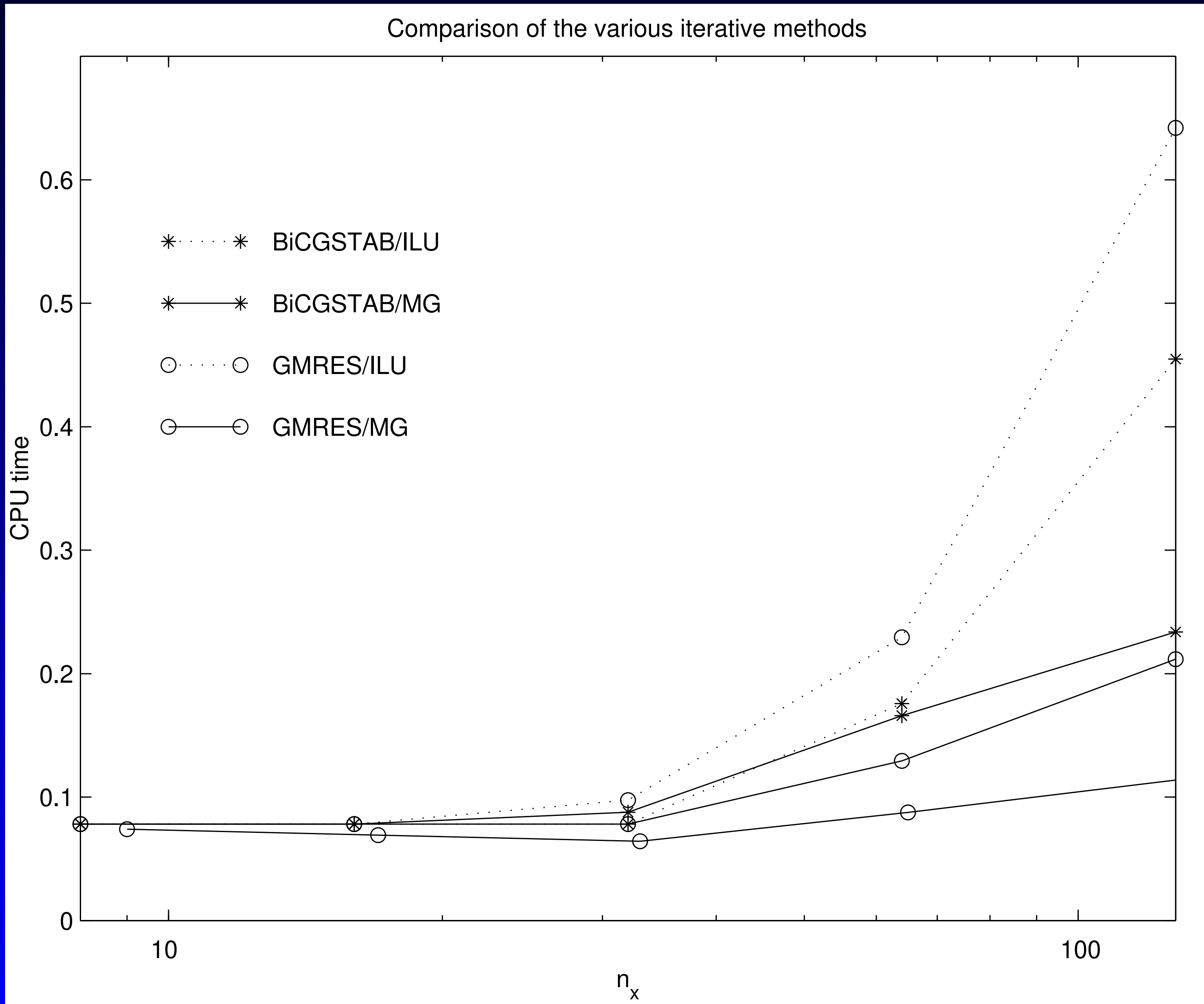
# Smoothers tested

- (Alternating) damped line Jacobi, 1 or 2 postsmoothing steps, no presmoothing
- Incomplete Block $LU$ decomposition, 1 postsmoothing step, no presmoothing





# Comparison



Comparison of the various iterative methods
BiCGSTAB/ILU
BiCGSTAB/MG
GMRES/ILU
GMRES/MG
CPU time
0
0.1
0.2
0.3
0.4
0.5
0.6
10
100
$n_x$

# Recursion

- Line Jacobi is recursive per line and can be massively parallelized, especially in 3D.
- What about IBLU? Classic IBLU is fully recursive.

# Divide and Conquer

The inversion of an $n \times n$ tridiagonal matrix can be executed in ${}^2\log n$ non recursive steps.

$$(I + LD^{-1} + UD^{-1})(D - L - U) =$$

$$D - LD^{-1}U - UD^{-1}L - LD^{-1}L - UD^{-1}U$$

$$= D_1 - L_1 - U_1$$

Bandwith is doubled in this operation.

TUDelft

# Incomplete Block Div and Conq

- Use the same formula, interpreted als blocks
- Use incomplete versions of $LD^{-1}U$ etc.
- $D$, $L$ and $U$ are (block)diagonals, consisting of tridiagonal blocks.
- We need 7 diagonals of $D^{-1}$
- Calculate from the productform of the inverse, keeping only 7 diagonals in ${}^2\log n$ steps

Recursion uses $O({}^2\log n)$ steps as claimed.