# Solution of systems of algebraic equations in CFD

Dr. Gergely Kristóf

26-th September 2009

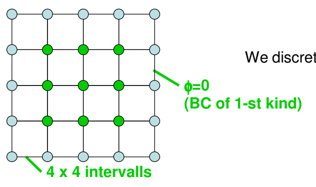
## The Poisson equation must be solved in every time step

in $\Psi$-$\omega$ method: $\Delta\psi = -\omega \longrightarrow \psi$

in pressure based methods: $\Delta P = \nabla\cdot \underline{f} \longrightarrow P$

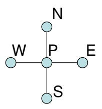
## A simple 2D example

The computational domain:



$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}=Q$$

We discretize this by using compass notations:

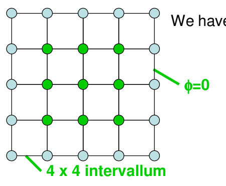


$$\frac{1}{\Delta x}\left(\frac{\phi_E-\phi_P}{\Delta x}-\frac{\phi_P-\phi_W}{\Delta x}\right)+\frac{1}{\Delta y}\left(\frac{\phi_N-\phi_P}{\Delta y}-\frac{\phi_P-\phi_S}{\Delta y}\right)=Q_P$$

On isotropic mesh: $\Delta x=\Delta y=h$ $\quad \phi_S+\phi_W-4\phi_P+\phi_E+\phi_N=h^2Q_P$

## In matrix form

$$\phi_S+\phi_W-4\phi_P+\phi_E+\phi_N=h^2Q_P$$



We have 9 unknowns.

$$A_{i,j}=\begin{bmatrix}-4&1&0&1&0&0&0&0&0\\1&-4&1&0&1&0&0&0&0\\0&1&-4&0&0&1&0&0&0\\1&0&0&-4&1&0&1&0&0\\0&1&0&1&-4&1&0&1&0\\0&0&1&0&1&-4&0&0&1\\0&0&0&1&0&0&-4&1&0\\0&0&0&0&1&0&1&-4&1\\0&0&0&0&0&1&0&1&-4\end{bmatrix}$$

The system now reads:

$$A_{i,j}\,\varphi_i = Q_i$$

The number of unknowns for 101x101 mesh N=$10^4$, therefore the number of elements of matrix A is $10^8$.

## Gauss elimination

As efficient as any other method for a general case, but it does not make use of the favorable characteristics of the matrix.

**1-st step Elimination:**

$$\begin{pmatrix}A_{1,1}&A_{1,2}&A_{1,3}\\A_{2,1}&A_{2,2}&A_{2,3}\\A_{3,1}&A_{3,2}&A_{3,3}\end{pmatrix}$$

$A_{21}/A_{11}$ times the first row is subtracted from the second row. The first element of the second row will become 0. Similarly, we eliminate the other elements of the second row up to the column N-1. } Repeated for every further rows.

**2-nd step Backsubstitution:**

$$\begin{pmatrix}U_{1,1}&U_{1,2}&U_{1,3}\\0&U_{2,2}&U_{2,3}\\0&0&U_{3,3}\end{pmatrix}$$

$$\phi_n=\frac{Q_n}{U_{nn}}$$

$$\phi_i=\frac{Q_i-\sum\limits_{k=k+1}^{N}U_{k,i}\,\phi_k}{U_{i,i}}$$

The operation cost of the method is $N^3/3$, out of which the back substitution requires only $N^2/2$ operations. Even if $A$ is sparse $U$ is not spares. The total memory requirement on a 2D mesh of 101x101 nodes is 400 Mb. **We don't need such an accurate solution because the discretization error is large anyway.**

## Iterative methods

The solution is refined step by step: approximation of $\phi$ in the n-th step is $\phi^n$.

By omitting the vector notations: $A\phi^n = Q-\rho^n$ $\quad$ **$\rho^n$ : residual**

The error: $\varepsilon^n=\phi-\phi^n$

$$A\varepsilon^n = A\left(\phi-\phi^n\right)=Q-\left(Q-\rho^n\right)=\rho^n$$

Thus, the same matrix describes the error.

Iterative methods: $M\,\phi^{n+1}=N\phi^n+Q$

For the converged solution: $\phi^{n+1}=\phi^n=\phi$ , therefore: $A=M-N$

Let's subtract $M\phi^n$ from both sides:

$$M\underbrace{\left(\phi^{n+1}-\phi^n\right)}_{\delta^n}=N\,\phi^n+Q-M\,\phi^n=Q-A\,\phi^n=\rho^n$$

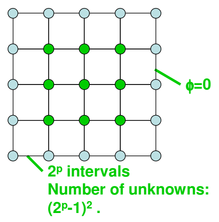
**correction:** $\delta^n$ $\quad M\,\delta^n=\rho^n$ $\quad$ This is the correction equation.

The better $M$ approximates $A$ is the faster the method converges.
$M$ must be easy to solve eg. diagonal, tri-diagonal, or a $\Delta$ matrix.

## Jacobi iteration

$$\phi_S^n+\phi_W^n-4\phi_P^{n+1}+\phi_E^n+\phi_N^n=h^2Q_P$$ $M$ is a diagonal matrix.

$$\phi_P^{n+1}=\frac{1}{4}\left(\phi_S^n+\phi_W^n+\phi_E^n+\phi_N^n-h^2Q_P\right)$$



Example program:

- The program...
- Major characteristics of the result...
- Required number of iterations ...

## Multigrid method

The correction equation for a simplified 1D problem:

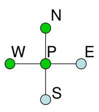
$$\frac{\partial^2\phi}{\partial x^2}=Q$$

$$\frac{1}{\Delta x^2}\left(\phi_{i-1}-2\phi_i+\phi_{i+1}\right)=Q_i$$

$$\frac{1}{\Delta x^2}\left(\phi_{i-1}^n-2\phi_i^n+\phi_{i+1}^n\right)=Q_i-\rho_i^n$$

$$\frac{1}{\Delta x^2}\left(\varepsilon_{i-1}^n-2\varepsilon_i^n+\varepsilon_{i+1}^n\right)=\rho_i^n$$

## Gauss-Seidel relaxation



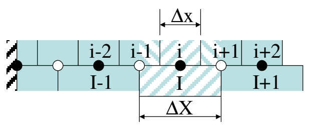
$$\phi_S^n+\underline{\phi_W^{n+1}}-4\phi_P^{n+1}+\phi_E^n+\underline{\phi_N^{n+1}}=h^2Q_P$$ M is a Δ matrix.

These terms are already known due to the calculation sequence

$$\phi_P^{n+1}=\frac{1}{4}\left(\phi_S^n+\phi_W^{n+1}+\phi_E^n+\phi_N^{n+1}-h^2Q_P\right)$$

- It requires halve as much iterations …
- and halve as much memory.
- The error is asymmetrically distributed.

We omit the iteration indices: $\frac{1}{\Delta x^2}\left(\varepsilon_{i-1}-2\varepsilon_i+\varepsilon_{i+1}\right)=\rho_i$



$$\frac{1}{\Delta x^2}\left(\frac{1}{2}\varepsilon_{i-2}\underline{-\varepsilon_{i-1}}+\frac{1}{2}\varepsilon_i\underline{+\varepsilon_{i-1}}-2\varepsilon_i\underline{+\varepsilon_{i+1}}+\frac{1}{2}\varepsilon_i\underline{-\varepsilon_{i+1}}+\frac{1}{2}\varepsilon_{i+2}\right)=$$

these terms are cancelled

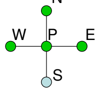
$$=\frac{1}{2}\rho_{i-1}+\rho_i+\frac{1}{2}\rho_{i+1}$$

$$\frac{1}{4\Delta x^2}\left(\varepsilon_{i-2}-2\varepsilon_i+\varepsilon_{i+2}\right)=\underbrace{\frac{1}{4}\left(\rho_{i-1}+2\rho_i+\rho_{i+1}\right)}_{\text{RESTRICTION}}$$

PROLONGATION

$$\frac{1}{\Delta X^2}\left(\varepsilon_{I-1}-2\varepsilon_I+\varepsilon_{I+1}\right)=\rho_I$$

## Line relaxation



$$\phi_S^n+\underline{\phi_W^{n+1}-4\phi_P^{n+1}+\phi_E^{n+1}}+\underline{\phi_N^{n+1}}=h^2Q_P$$

these values are known

these obtained line by line from the tri-diagonal system solved by the Thomas algorithm.

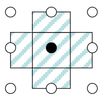
Note - Much more efficient methods based on the tri-diagonal solver also exist: Operator Splitting (or Alternating Direction Implicit, ADI) methods.

The problem:
The above mentioned methods are only **smoothing** the solution.
The boundary effects need a very long time to penetrate the computational domain.

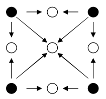
Solution:
We need to use coarser meshes too. The first estimates of the correction can be obtained on a coarser mesh, than can be refined on the fine mesh.

## Generalization to 2D or 3D:

Restriction:



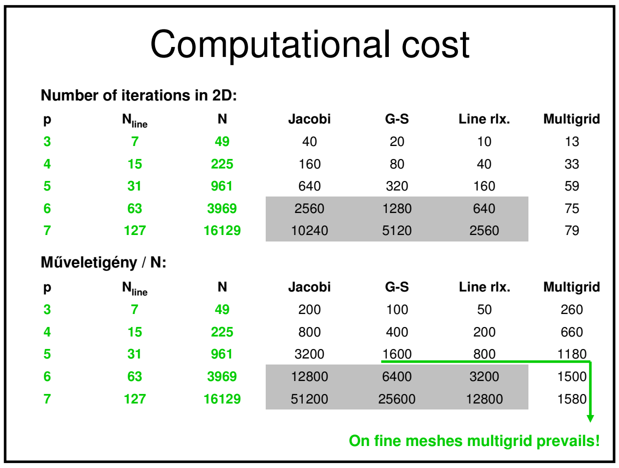
Prolongation:



1. Restriction: $\rho_i \rightarrow \rho_I$
2. Calculation of $\varepsilon_I$ . Eg. in 3D we have an 8 fold reduced number of unknowns.
3. Prolongation of $\varepsilon_I$ to the fine mesh. ($\varepsilon_i$),
4. Smoothing on the fine mesh.

Why shouldn't we use an even more coarse mesh when calculating $\varepsilon_I$ ?

1. Evaluation of the residuals on the finest mesh.
2. Consecutive restrictions of $\rho$ to every coarser mesh.
3. Solution of the system on the coarsest mesh. (Even by using a direct method.)
4. Consecutively for every finer mesh:
   - Prolongation of $\varepsilon$
   - Smoothing (Eg. by using Gauss-Seidel relaxation.)
5. Correction of $\phi$. (Only on the finest mesh).



## Computational cost

**Number of iterations in 2D:**

| p | $N_{line}$ | N | Jacobi | G-S | Line rlx. | Multigrid |
|---|---|---|---|---|---|---|
| 3 | 7 | 49 | 40 | 20 | 10 | 13 |
| 4 | 15 | 225 | 160 | 80 | 40 | 33 |
| 5 | 31 | 961 | 640 | 320 | 160 | 59 |
| 6 | 63 | 3969 | 2560 | 1280 | 640 | 75 |
| 7 | 127 | 16129 | 10240 | 5120 | 2560 | 79 |

**Műveletigény / N:**

| p | $N_{line}$ | N | Jacobi | G-S | Line rlx. | Multigrid |
|---|---|---|---|---|---|---|
| 3 | 7 | 49 | 200 | 100 | 50 | 260 |
| 4 | 15 | 225 | 800 | 400 | 200 | 660 |
| 5 | 31 | 961 | 3200 | 1600 | 800 | 1180 |
| 6 | 63 | 3969 | 12800 | 6400 | 3200 | 1500 |
| 7 | 127 | 16129 | 51200 | 25600 | 12800 | 1580 |

**On fine meshes multigrid prevails!**