Integration of Linear Elliptic Equations

- **Poisson's Equation**

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x,y) \qquad (1)$$

$(x,y) \in \Omega = [0,1] \times [0,1]$ "Unit Square"

Equation (1) is subject to Dirichlet boundary conditions

$$u(x,y) = g(x,y) \;\; ; \;\; (x,y) \in \partial\Omega \qquad (2)$$

→ For $f = 0$, "Laplace's Equation" is recovered.

- Many physical phenomena are described by equations (1), (2):
  - Steady incompressible inviscid flow
  - Static deformation of a membrane under distributed loads
  - Equilibrium temperature in a square slab.

- For equation (1), the independent variables x & y are space-like, and the problem is independent of time.
- The general exact solution of (1) involves complex variables and conformal mapping techniques that become quite involved except for some special cases of $g(x,y)$ in (2).

Numerical Solution:

- Let us define a mesh system

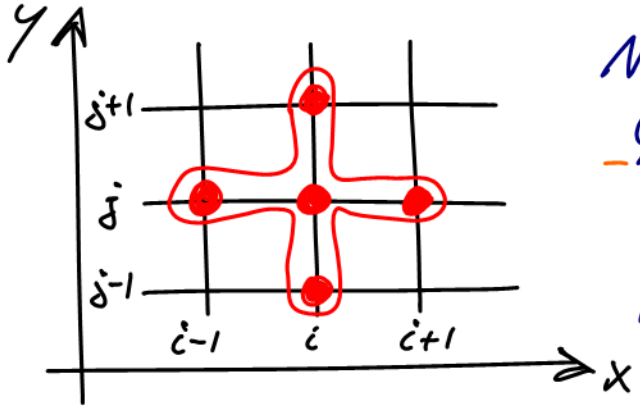
$$\begin{cases} x_i = (i-1)\,\Delta x \quad ; \quad \Delta x = \dfrac{1}{ix-1} \\ y_j = (j-1)\,\Delta y \quad ; \quad \Delta y = \dfrac{1}{jx-1} \end{cases} \tag{3}$$

- A numerical scheme can be devised as:

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} = f_{i,j} \tag{4}$$

$$i = 2, \dots, ix-1 \quad ; \quad j = 2, \dots, jx-1$$

- This is the classical "5-Point Scheme".



Note that the entire Computational Molecule is unknown as the problem is time independent.

- Truncation Error:

$$\varepsilon_{i,j} = \cancel{\frac{\partial^2 u_{i,j}}{\partial x^2}} + \frac{\Delta x^2}{4!}\frac{\partial^4 u_{i,j}}{\partial x^4} + O(\Delta x^4)$$

$$+ \cancel{\frac{\partial^2 u_{i,j}}{\partial y^2}} + \frac{\Delta y^2}{4!}\frac{\partial^4 u_{i,j}}{\partial y^4} + O(\Delta y^4)$$

$$\varepsilon_{i,j} = O(\Delta x^2, \Delta y^2)$$

$\Rightarrow$ The scheme (4) is consistent and 2nd order accurate in x and y.

- How about stability?

So far, we have not defined any time-like (marching) direction. In fact, we cannot,

as an elliptic equation only has imaginary characteristics.

- So we consider all computational molecules described by (4) as a plane of unknowns $u_{i,j}$ except at the boundaries of Dirichlet type.

$$\begin{cases} u_{i,1} = g_{i,1} \\ u_{i,jx} = g_{i,jx} \end{cases} ; \; i = 1, \dots, ix$$

$$\begin{cases} u_{1,j} = g_{1,j} \\ u_{ix,j} = g_{ix,j} \end{cases} ; \; j = 1, \dots, jx \qquad (5)$$

- Matrix Formulation:

Equation (4) describes a linear algebraic system for the unknowns $u_{i,j}$. Let us order $u_{i,j}$ in a vector of unknowns such that the j-index is iterated before the i-index.

$$\underline{u} = \begin{bmatrix} u_{1,1} \\ u_{1,2} \\ \dots \\ u_{1,jx} \\ \dots \\ u_{i,1} \\ \dots \\ u_{ix,jx} \end{bmatrix}$$

- Now we can write the linear system as

$$\boxed{A \cdot \underline{u} = \underline{b}} \quad (6)$$

where the matrix A has the following structure.

$$\boxed{A = \left[\dots 0 \ \frac{1}{\Delta x^2} \ 0 \dots 0 \ \frac{1}{\Delta y^2} \ -\frac{1}{h^2} \ \frac{1}{\Delta y^2} \ 0 \dots 0 \ \frac{1}{\Delta x^2} \ 0 \dots \right]} \quad (7)$$

where $\underline{\frac{1}{h^2} = 2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)}$

- It is clear that A is a pentadiagonal matrix. The vector $\underline{b}$ contains the r.h.s. in (4) and the boundary conditions in (5).

- The matrix A is a large $(ix * jx)^2$ sparse matrix with entries along the diagonals. The structure is a consequence of the computational molecule.

- The obvious solution to (6) is $\underline{u} = A^{-1} \cdot \underline{b}$.
  The inverse matrix A^{-1} is very expensive to compute with lots of storage requirement.
- Direct Methods do not require to compute A^{-1}.

  Examples:

  → Gramer's Rule

  The algorithm becomes very time consuming for $ix * jx > 3$, as the number of operations is proportional to $(ix * (jx+1))!$ ⟹ Do not use.

  → Gauss Elimination

  This is a useful and efficient tool in particular for sparse matrices. It requires approximately $(ix * jx)^3$ operations.

- However, the solution of large systems of

algebraic equations is performed most efficiently with iterative methods.

- Iterative Methods

  Examples:

  → Jacobi Method

  → Gauss-Seidel Method

  → Over-Relaxation Method

The basic idea of the iterative method is to construct a sequence of approximations u^n that are being marched in an imaginary time direction.

u^n ; $n \hat{=}$ Iteration (Time) Step

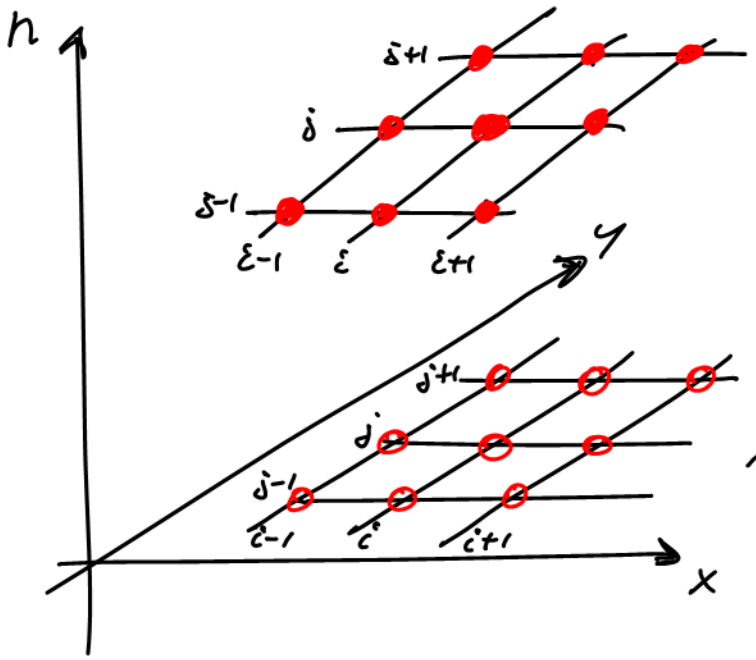
u^0 represent an arbitrary initial solution vector. Next we consider so-called two-level methods. This means that "new"

values $\underline{u}^{n+1}$ depend only on the "old" values $\underline{u}^{n}$. These methods can be represented as

$$\underline{u}^{n+1} - \underline{u}^{n} = H\,(A.\underline{u}^{n} - \underline{b}) \qquad (8)$$

where H is a conditioning matrix induced by the iterative method. From (8) it is apparent that equation (6) is satisfied for $\underline{u}^{n+1} - \underline{u}^{n} \to 0$.

↦ How does the "imaginary" time marching work?



"Unknown" Solution Plane @ iteration $n+1$

Solution Plane @ iteration n

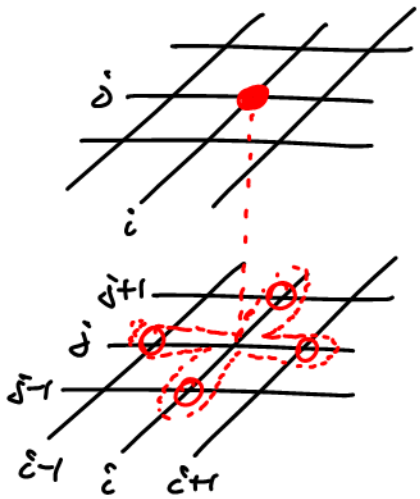
Iterative Methods "Jacobi Method"

$$\frac{u^{n}_{i+1,j} - 2u^{n+1}_{i,j} + u^{n}_{i-1,j}}{\Delta x^2} + \frac{u^{n}_{i,j+1} - 2u^{n+1}_{i,j} + u^{n}_{i,j-1}}{\Delta y^2} = f_{i,j} \qquad (9)$$

$$i = 2, \dots, ix-1 \quad ; \quad j = 2, \dots, jx-1$$

- The "new" value for the unknown $u^{n+1}_{i,j}$ is obtained from the central term of the scheme.

Computational Molecule



Iteration $n+1$

Iteration n

- We know through (4) that the scheme (9) is consistent and 2nd order accurate in x and y.

But ... is it also "stable"?

As we defined now an imaginary marching direction, we have to study the stability of scheme (9). Once more we use v. Neumann analysis and write (9) in its homogeneous form ($f_{i,j}=0$) in update form as:

$$\frac{1}{h^2}\cdot u_{i,j}^{n+1} = \frac{1}{\Delta x^2}\left(u_{i+1,j}^{n}+u_{i-1,j}^{n}\right)+\frac{1}{\Delta y^2}\left(u_{i,j+1}^{n}+u_{i,j-1}^{n}\right) \tag{9a}$$

We introduce the complex Fourier mode as

$$u_{i,j}^{n} = g^n e^{\xi i\alpha} e^{\xi j\beta} \tag{10}$$

Note in (10) that none of the independent variables x and y is time-like ..... again ... because it's an elliptic equation with no real characteristic. $\Longrightarrow$

$$g^{n+1} e^{\xi i\alpha} e^{\xi j\beta} = \frac{h^2}{\Delta x^2}\left(g^n e^{\xi(i+1)\alpha} e^{\xi j\beta} + g^n e^{\xi(i-1)\alpha} e^{\xi j\beta}\right)$$

$$+\frac{h^2}{\Delta y^2}\left(g^n e^{\xi i\alpha} e^{\xi(j+1)\beta} + g^n e^{\xi i\alpha} e^{\xi(j-1)\beta}\right)$$

Divide by $u_{i,j}^n = g^n e^{\underline{i}i\alpha} e^{\underline{i}j\beta}$

$$g = \frac{h^2}{\Delta x^2}\left(e^{\underline{i}\alpha} + e^{\underline{i}(-\alpha)}\right) + \frac{h^2}{\Delta y^2}\left(e^{\underline{i}\beta} + e^{\underline{i}(-\beta)}\right)$$

$$g = \frac{h^2}{\Delta x^2}\left(\cos\alpha + \underline{i}\sin\alpha + \cos(-\alpha) + \underline{i}\sin(-\alpha)\right)$$

$$+ \frac{h^2}{\Delta y^2}\left(\cos\beta + \underline{i}\sin\beta + \cos(-\beta) + \underline{i}\sin(-\beta)\right)$$

$$g = \frac{2h^2}{\Delta x^2}\cos\alpha + \frac{2h^2}{\Delta y^2}\cos\beta \qquad \text{"real"}$$

$$\|g\| = \frac{2h^2}{\Delta x^2}\|\cos\alpha\| + \frac{2h^2}{\Delta y^2}\|\cos\beta\| \qquad (11)$$

$$\leq \frac{2h^2}{\Delta x^2} + \frac{2h^2}{\Delta y^2} = 2\left(\frac{\frac{1}{\Delta x^2}}{\frac{1}{h^2}} + \frac{\frac{1}{\Delta y^2}}{\frac{1}{h^2}}\right)$$

$$= \frac{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}}{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}} = 1 \leq 1$$

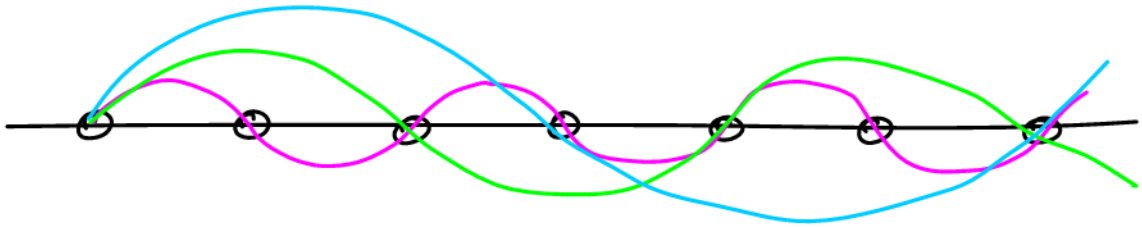
$\Rightarrow$ The Jacobi Method (9) is unconditionally stable.

- Some comments on "Convergence Speed":

  → The convergence speed is governed by the largest value of $\|g\|$.

  → Note that a given mesh can support wave modes that satisfy

$$\alpha = k \cdot \pi \Delta x \quad ; \quad k = 1, \dots, jx-1$$
$$\beta = l \cdot \pi \Delta y \quad ; \quad l = 1, \dots, jx-1 \qquad (12)$$



  → The convergence speed is determined by the largest possible complex amplitude $\|g\|_{max}$.

  → With reference to (11), this occurs for the lowest wave mode possible on a given mesh, i.e.

$$\|g\|_{max} = 2h^2 \left(\frac{\cos(\pi \Delta x)}{\Delta x^2} + \frac{\cos(\pi \Delta y)}{\Delta y^2} \right) \qquad (13)$$

Let us expand the cosine function as a Taylor series

$$\cos(\pi \Delta x) = \cos(0) + \pi \Delta x \,(-\sin(0)) + \frac{(\pi \Delta x)^2}{2}(-\cos(0)) + \mathcal{O}(\Delta x^3)$$
$$= 1 - \tfrac{1}{2}\pi^2 \Delta x^2 + \mathcal{O}(\Delta x^3)$$

$$\cos(\pi \Delta y) = \cos(0) + \pi \Delta y \,(-\sin(0)) + \frac{(\pi \Delta y)^2}{2}(-\cos(0)) + \mathcal{O}(\Delta y^3)$$
$$= 1 - \tfrac{1}{2}\pi^2 \Delta y^2 + \mathcal{O}(\Delta y^3)$$

Hence we can estimate $\|g\|_{max}$ in (13) as

$$\|g\|_{max} \approx 2h^2 \left[\frac{1}{\Delta x^2}\left(1 - \tfrac{1}{2}\pi^2 \Delta x^2\right) + \frac{1}{\Delta y^2}\left(1 - \tfrac{1}{2}\pi^2 \Delta y^2\right) \right]$$

$$= 2h^2 \Big[\underbrace{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}}_{= \frac{1}{2}\frac{1}{h^2}} - \pi^2 \Big]$$

$$\Rightarrow \quad \boxed{\|g\|_{max} \approx 1 - 2\pi^2 h^2} \qquad (14)$$

After (N) iteration steps, the error amplitude is reduced to $\|g\|_{max}^N$. Therefore, the smaller

$\|g\|_{max}$, the faster convergence will be.

For the Jacobi Method, $\|g\|_{max}$ is governed by (14) and close to 1. In fact, $\|g\|_{max} \to 1$ as $h \to 0$.

$\Rightarrow$ The Jacobi Method is expected to be very slow on fine meshes !

The exponential behavior of the error amplitude $\|g\|_{max}^{N}$ over the iterations suggest a definition of a Convergence Rate as

$$R = -\ln \|g\|_{max} \qquad (15)$$

Using (14) for the Jacobi Method, we obtain

$$R = -\ln\left(1 - 2\pi^2 h^2\right)$$

$$\approx -\left[\cancel{\ln(1)} - 2\pi^2 h^2 \frac{1}{1} + O(h^4)\right]$$

$$R \approx 2\pi^2 h^2 \to 0 \quad \text{for} \quad h \to 0$$

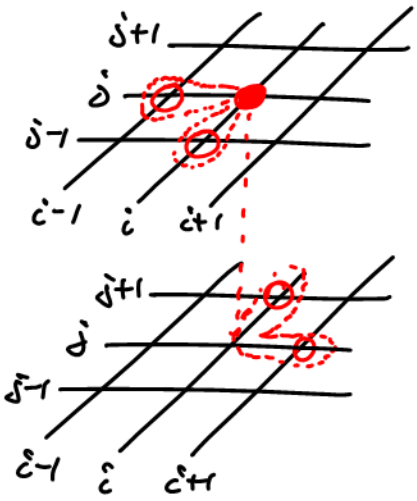
Iterative Methods "Gauss-Seidel Method"

$$\frac{u^{n}_{i+1,j} - 2u^{n+1}_{i,j} + u^{n+1}_{i-1,j}}{\Delta x^2} + \frac{u^{n}_{i,j+1} - 2u^{n+1}_{i,j} + u^{n+1}_{i,j-1}}{\Delta y^2} = f_{i,j} \qquad (15)$$

$i = 2, \dots, ix-1 \qquad j = 2, \dots, jx-1$

- Comparison of (15) with (9) reveals that now values of $u_{i-1,j}$ and $u_{i,j-1}$ are also at the new iteration (time) step.

Computational Molecule



Iteration (n+1)

Iteration (n)

} Sweep grid points as $i = 2, \dots, ix-1$, $j = 2, \dots, jx-1$

- However, the values $u^{n+1}_{i-1,j}$ and $u^{n+1}_{i,j-1}$ are **not** unknown at iteration n+1 if the indices i and j are swept through with increasing numbers.

- The fact that "updated" values are used rapidly in the solution algorithm lets the Gauss-Seidel Method converge faster than the Jacobi Method.
- Also, the Gauss-Seidel Method does not require any additional storage than the basic arrays.
- The method is a special case of the more general over-relaxation method.

Iterative Methods "Successive Over-Relaxation Method (SOR)"

$$\frac{u^{n}_{i+1,j} - 2\tilde{u}_{i,j} + u^{n+1}_{i-1,j}}{\Delta x^2} + \frac{u^{n}_{i,j+1} - 2\tilde{u}_{i,j} + u^{n+1}_{i,j-1}}{\Delta y^2} = f_{i,j} \qquad (16)$$

$$i = 2, \dots, i_x - 1 \qquad j = 2, \dots, j_x - 1$$

- Equation (16) presents a FD scheme to solve for a provisional "new" value $\tilde{u}_{i,j}$. The actual "new" value $u^{n+1}_{i,j}$ is obtained from

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \omega \left(\tilde{u}_{i,j} - u_{i,j}^{n} \right) \qquad (17)$$

where ω is a relaxation factor. It is clear that the Gauss-Seidel Method is recovered for $\omega = 1$.

→ Over-Relaxation $(\omega > 1)$

The value for $u_{i,j}^{n+1}$ experiences a larger change than expected with Gauss-Seidel.

⟹ Convergence Acceleration

→ Under-Relaxation $(0 < \omega < 1)$

For certain governing equations, the convergence rate may have to be decreased in order to maintain numerical stability.

- Solving (17) for $\tilde{u}_{i,j}$ yields

$$\tilde{u}_{i,j} = u_{i,j}^{n} + \frac{1}{\omega} \left(u_{i,j}^{n+1} - u_{i,j}^{n} \right) \qquad (18)$$

and using (18) in (16) results in

$$\frac{u^n_{i+1,j} - 2\left[u^n_{i,j} + \frac{1}{\omega}\left(u^{n+1}_{i,j} - u^n_{i,j}\right)\right] + u^{n+1}_{i-1,j}}{\Delta x^2} + \frac{u^n_{i,j+1} - 2\left[u^n_{i,j} + \frac{1}{\omega}\left(u^{n+1}_{i,j} - u^n_{i,j}\right)\right] + u^{n+1}_{i,j-1}}{\Delta y^2} = f_{i,j} \tag{19}$$

- The scheme (19) is still 2nd order accurate in the spatial directions x and y.

⟶ But how about stability? ($f_{i,j} = 0$)

Write (19) in "update" form ...

$$\frac{1}{\omega}\frac{1}{h^2} u^{n+1}_{i,j} = \frac{1}{\Delta x^2}\left(u^n_{i+1,j} + u^{n+1}_{i-1,j}\right) + \frac{1}{\Delta y^2}\left(u^n_{i,j+1} + u^{n+1}_{i,j-1}\right) - \frac{1}{h^2}\left(1 - \frac{1}{\omega}\right) u^n_{i,j} \tag{20}$$

- Introduce the complex Fourier mode (10)

$$u^n_{i,j} = g^n e^{\mathrm{i}i\alpha} e^{\mathrm{i}j\beta}$$

$$\frac{1}{\omega}\frac{1}{h^2} g^{n+1} e^{\xi i\alpha} e^{\xi i\beta}$$

$$= \frac{1}{\Delta x^2}\left(g^n e^{\xi(i+1)\alpha} e^{\xi i\beta} + g^{n+1} e^{\xi(i-1)\alpha} e^{\xi i\beta}\right)$$

$$+ \frac{1}{\Delta y^2}\left(g^n e^{\xi i\alpha} e^{\xi(i+1)\beta} + g^{n+1} e^{\xi i\alpha} e^{\xi(i-1)\beta}\right)$$

$$- \frac{1}{h^2}\left(1 - \frac{1}{\omega}\right) g^n e^{\xi i\alpha} e^{\xi i\beta}$$

Divide by $u^n_{ij} = g^n e^{\xi i\alpha} e^{\xi i\beta}$

$$\frac{1}{h^2}\frac{1}{\omega} g = \frac{1}{\Delta x^2}\left(e^{\xi\alpha} + g e^{\xi(-\alpha)}\right) + \frac{1}{\Delta y^2}\left(e^{\xi\beta} + g e^{\xi(-\beta)}\right)$$

$$- \frac{1}{h^2}\left(1 - \frac{1}{\omega}\right)$$

$$\left(\frac{1}{h^2\omega} - \frac{1}{\Delta x^2} e^{\xi(-\alpha)} - \frac{1}{\Delta y^2} e^{\xi(-\beta)}\right) * g$$

$$= \frac{1}{\Delta x^2} e^{\xi\alpha} + \frac{1}{\Delta y^2} e^{\xi\beta} - \frac{1}{h^2}\left(1 - \frac{1}{\omega}\right)$$

$$\left[\frac{1}{h^2\omega} - \frac{1}{\Delta x^2}\left(\cos(-\alpha) + \xi \sin(-\alpha)\right) - \frac{1}{\Delta y^2}\left(\cos(-\beta) + \xi \sin(-\beta)\right)\right] g$$

$$= \frac{1}{\Delta x^2}\left(\cos\alpha + \xi \sin\alpha\right) + \frac{1}{\Delta y^2}\left(\cos\beta + \xi \sin\beta\right) - \frac{1}{h^2}\left(1 - \frac{1}{\omega}\right)$$

- We can rearrange the above expression as

$$\left[\frac{1}{4^2\omega} - \frac{\cos\alpha}{\Delta x^2} - \frac{\cos\beta}{\Delta y^2} + \underline{i}\left(\frac{\sin\alpha}{\Delta x^2} + \frac{\sin\beta}{\Delta y^2}\right)\right] g$$

$$= \frac{\cos\alpha}{\Delta x^2} + \frac{\cos\beta}{\Delta y^2} - \frac{1}{h^2} + \frac{1}{h^2\omega} + \underline{i}\left(\frac{\sin\alpha}{\Delta x^2} + \frac{\sin\beta}{\Delta y^2}\right) \qquad (21)$$

- Let's remember that we would like to investigate the condition $\|g\| \leq 1$. We therefore write (21) in a more suitable form as

$$\left[\frac{1-\cos\alpha}{\Delta x^2} + \frac{1-\cos\beta}{\Delta y^2} + \frac{2-\omega}{2h^2\omega} + \underline{i}\left(\frac{\sin\alpha}{\Delta x^2} + \frac{\sin\beta}{\Delta y^2}\right)\right] g$$

$$= -\underbrace{\left(\frac{1-\cos\alpha}{\Delta x^2} + \frac{1-\cos\beta}{\Delta y^2}\right)}_{a} - \underbrace{\left(\frac{2-\omega}{2h^2\omega}\right)}_{b} + \underline{i}\underbrace{\left(\frac{\sin\alpha}{\Delta x^2} + \frac{\sin\beta}{\Delta y^2}\right)}_{c} \qquad (22)$$

- We write (22) as

$$\left[(a+b) + \underline{i}\,c\right] g = -(a-b) + \underline{i}\,c$$

$$\|(a+b) + \underline{i}\,c\| * \|g\| = \|-(a-b) + \underline{i}\,c\|$$

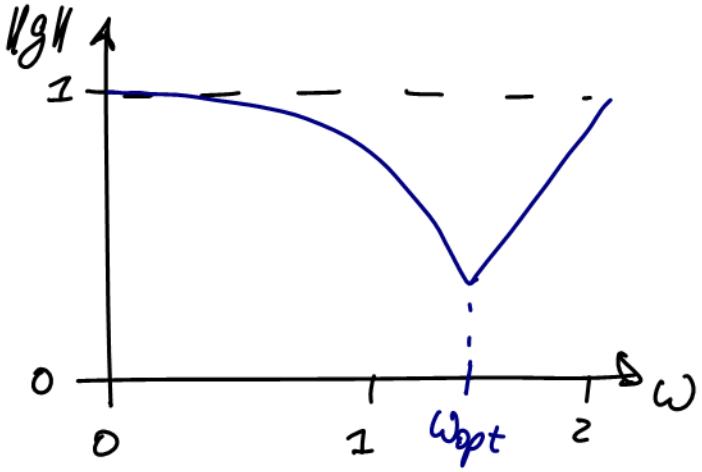
$$\left[(a+b)^2 + c^2\right] \|g\| = (a-b)^2 + c^2$$

$\Longrightarrow$ Now we can find an expression for $\|g\|$

$$\|g\| = \frac{(a-b)^2+c^2}{(a+b)^2+c^2} \overset{!}{\leq} 1 \qquad (23)$$

$$b = \frac{2-\omega}{2h^2\omega} \geq 0 \Longrightarrow \omega < 2 \qquad (24)$$

- The SOR Method is unconditionally stable for $0 < \omega < 2$. This includes the special case of the Gauss-Seidel Method with $\omega = 1$.
- Some comments on "Convergence Speed"
  - → For each ω, there is a maximum value of $\|g(\omega)\|$ corresponding to $\alpha = \pi \cdot \Delta x$ and $\beta = \pi \cdot \Delta y$ (lowest wave numbers).
  - → Convergence speed will have a maximum for the lowest value of $\|g(\omega)\|$.



Which ω_{opt}?

In a general case, the best over-relaxation factor ω_{opt} has to be found by trial & error. $\omega_{opt} = 1.8$

- Let us find a Convergence Rate (R) for the SOR Method. First, we relate (ω) to a parameter describing the grid spacing, i.e.

$$\frac{1}{h^2} = 2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right) \quad \text{via} \quad \boxed{\omega = 2 - c \cdot h} \tag{25}$$

where $c = \text{const.}$ and $\omega \mapsto 2$ for $h \mapsto 0$.

- It can be shown after some algebra that

$$\boxed{\|g(\omega_{opt})\| = 1 - \pi h + O(h^2)} \tag{26}$$

- Hence the convergence rate R becomes

$$\boxed{R = -\ln \|g(\omega_{opt})\| \approx \pi h} \tag{27}$$

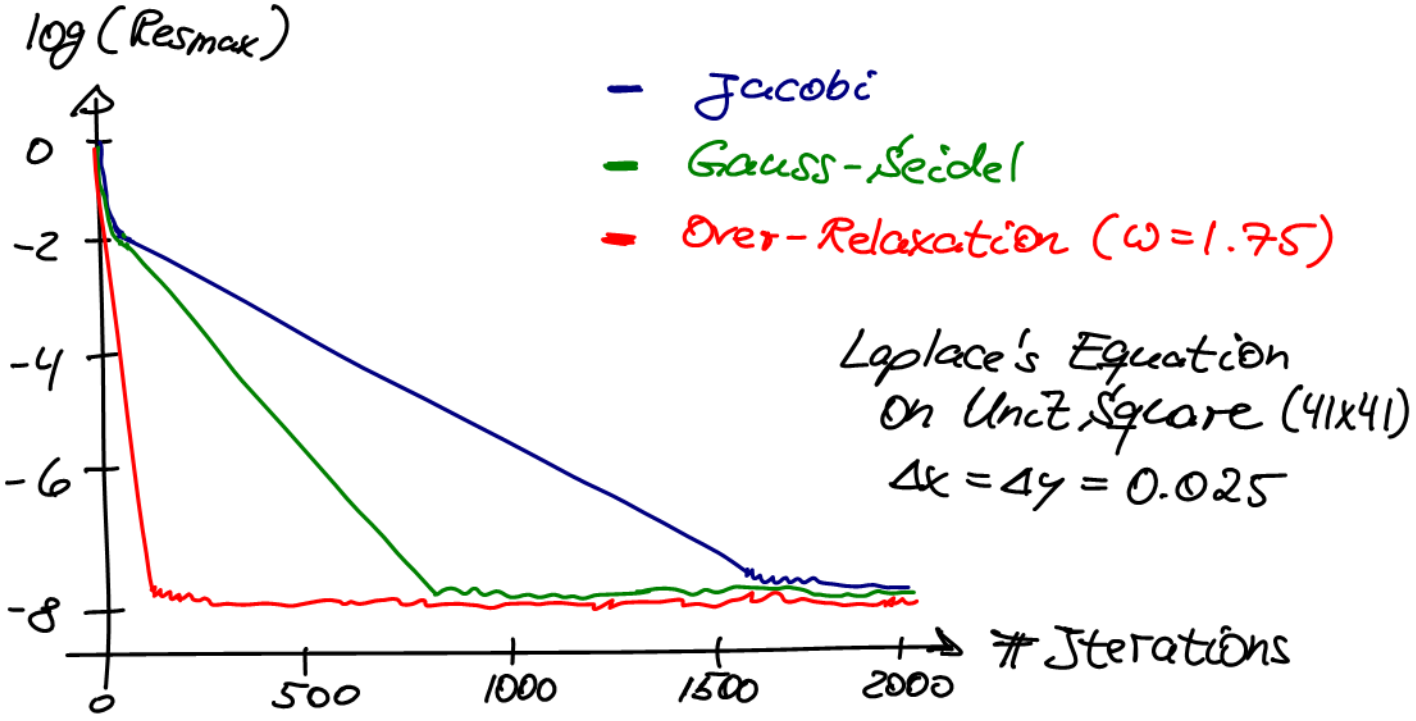
- Therefore, the SOR Method converges one order of magnitude faster than the Jacobi Method in (15).

- It can be shown that the Gauss-Seidel Method converges twice as fast as the Jacobi Method.

- Let us define a Residual (Res) with reference to (4) and the truncation error TE as

$$Res(n) = \frac{u^n_{i+1,j} - 2u^n_{i,j} + u^n_{i-1,j}}{\Delta x^2} + \frac{u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1}}{\Delta y^2} - f_{i,j}$$

The maximum Residual (Res_{max}) is a measure for how well a solution is converged. (28)



- The theoretical results are recovered quite well.
  - i) Convergence Rate of Jacobi Method $\approx 2\pi^2 h^2$
  - ii) Gauss-Seidel twice as fast as Jacobi
  - iii) Convergence Rate of Over-Relaxation $\approx \pi h$

Iterative Methods — "Successive Line Over-Relaxation (SLOR)"

- SOR has been found superior to Jacobi and Gauss-Seidel. Block iterative methods can exhibit faster convergence than SOR. Here subgroups of unknowns are solved simultaneously along an entire grid line using the efficient Thomas algorithm. → Successive Line Over-Relaxation (SLOR)

$$\frac{u^{n}_{i+1,j} - 2\tilde{u}_{i,j} + u^{n+1}_{i-1,j}}{\Delta x^2} + \frac{\tilde{u}_{i,j+1} - 2\tilde{u}_{i,j} + \tilde{u}_{i,j-1}}{\Delta y^2} = f_{i,j} \qquad (29)$$

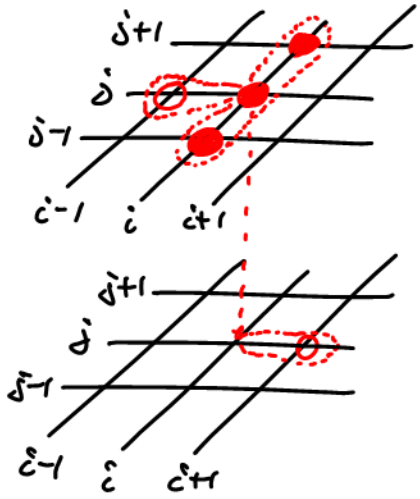
- Note in (29) that the scheme is implicit-like along a j-line of constant i.

- The "updated" value $u^{n+1}_{i,j}$ is obtained from

$$u^{n+1}_{i,j} = u^{n}_{i,j} + \omega\left(\tilde{u}_{i,j} - u^{n}_{i,j}\right) \qquad (30)$$

after the entire line $\tilde{u}_{i,j}$ $(i = \text{const.},\ j = 2, \ldots, jx-1)$ has been solved using the Thomas algorithm.

Computational Molecule



Iteration (n+1)

Iteration (n)

Sweep grid lines as $i = 2, \ldots, ix-1$

Solve i-line for $j = 2, \ldots, jx-1$

- It can be shown that the relaxation factor (ω) is subject to to a stability restriction of $0 < \omega < 2$.

- The SLOR method requires more operations per sweep than the SOR method. In general, however, it is less expensive, as it requires fewer iteration sweeps. This is due to the fact that the boundary conditions at $j = 1$ and $j = jx$ are being felt immediately along the i-line.

Iterative Methods "Alternate Direction Implicit (ADI)"

- The SLOR method introduced the idea of making the iterative process feel the boundary conditions faster in order to reduce the required number of iteration sweeps.
- The basic procedure was to make the computational molecule implicit-like in one spatial direction.
- In an ADI method, the same idea is used, however (as the name says it!) the implicit-like direction is altered to have the evolving solution feel (all) boundary conditions of the elliptic system.
- This is achieved using an intermediate time (iteration) step.
- A two-step Peaceman-Rachford ADI scheme for Poisson's equation reads:

Step 1:

(31)

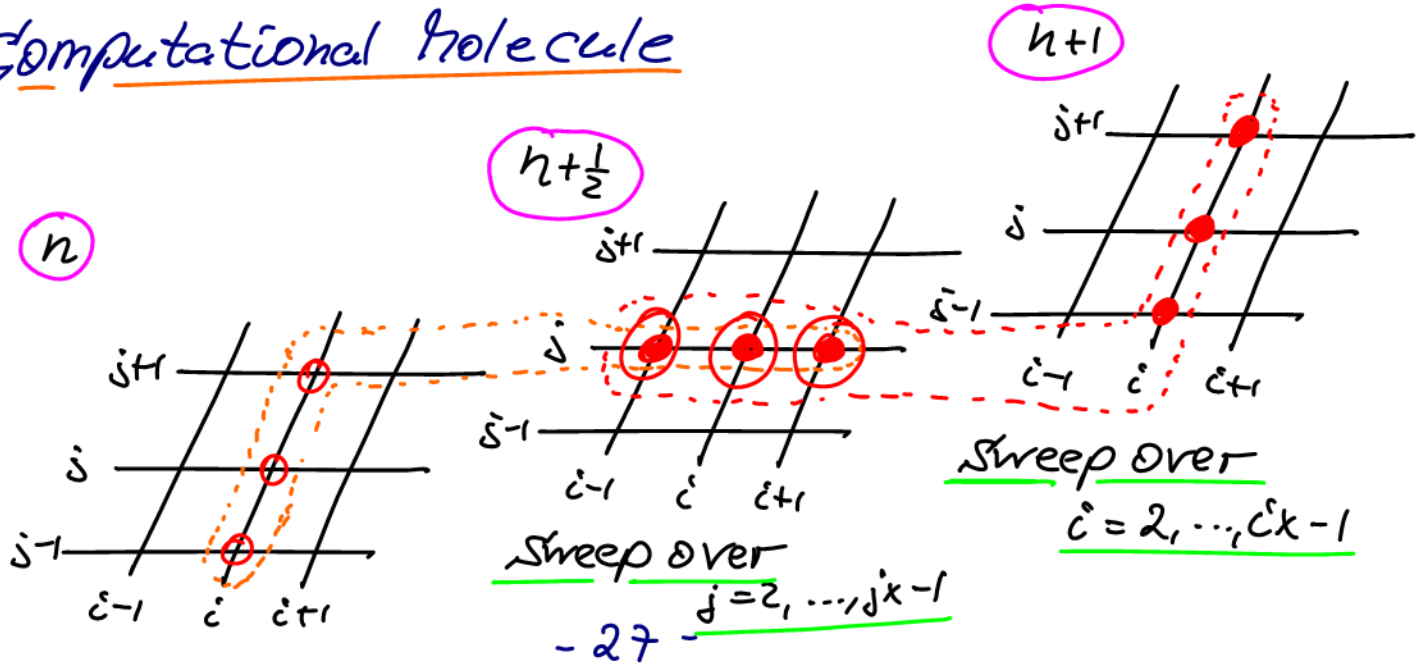
$$u_{i,j}^{n+\frac{1}{2}} = u_{i,j}^{n} + \delta^{n}\left(\frac{u_{i+1,j}^{n+\frac{1}{2}} - 2u_{i,j}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}}}{\Delta x^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}}{\Delta y^2} - f_{i,j}\right)$$

Step 2:

(32)

$$u_{i,j}^{n+1} = u_{i,j}^{n+\frac{1}{2}} + \delta^{n}\left(\frac{u_{i+1,j}^{n+\frac{1}{2}} - 2u_{i,j}^{n+\frac{1}{2}} + u_{i-1,j}^{n+\frac{1}{2}}}{\Delta x^2} + \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{\Delta y^2} - f_{i,j}\right)$$

Computational Molecule



- The (s^n) are iteration parameters. For the Laplace equation ($f_{i,j} = 0$), the Peaceman-Rachford iteration procedure is convergent for a fixed value of s^n.

- For maximum computational efficiency, however, the iteration parameters should be cycled with (n). The key to a successful ADI implementation for elliptic equations lies in the proper choice of (s^n)'s.

- If $b_x = s^n/\Delta x^2$ and $b_y = s^n/\Delta y^2$, it can be shown that the complex amplification amplitudes for steps 1 and 2 are

$$g_1 = \frac{1 - 2b_y(1-\cos\beta)}{1 + 2b_x(1-\cos\alpha)} \quad ; \quad g_2 = \frac{1 - 2b_x(1-\cos\alpha)}{1 + 2b_y(1-\cos\beta)} \tag{33}$$

- It is apparent from (33) that the total amplification factor $g = g_1 * g_2$ satisfies $\|g\| \leq 1 \quad \forall \alpha, \beta \Rightarrow$ The ADI method is unconditionally stable!