Solving elliptic PDEs

Mostly consider the Poisson equation $\nabla^2 u(\vec{r}) = -f(\vec{r})$

Solve it in some region of space Ω. On the boundaries Γ, boundary conditions:

$$u(\vec{r})\big|_\Gamma = g(\vec{r}) \qquad \text{(Dirichlet)}$$

or

$$\vec{n}\cdot\nabla u(\vec{r})\big|_\Gamma \equiv \left.\frac{\partial u(\vec{r})}{\partial n}\right|_\Gamma = g(\vec{r}) \qquad \text{(Neumann)}$$

Can be a mixture: Dirichlet on some part and Neumann on the rest.

$$\nabla^2 u(\vec{r}) = -f(\vec{r})$$

$$u(\vec{r})\big|_\Gamma = g(\vec{r}) \text{ (Dirichlet)} \quad \text{or} \quad \vec{n}\cdot\nabla u(\vec{r})\big|_\Gamma = g(\vec{r}) \text{ (Neumann)}$$

E.g., electrostatics:

$$\nabla^2 \phi(\vec{r}) = -\frac{4\pi\rho(\vec{r})}{\epsilon} \text{ (CGS)} \quad \text{or} \quad \nabla^2 \phi(\vec{r}) = -\frac{\rho(\vec{r})}{\epsilon_0 \epsilon} \text{ (SI)}$$

Here assume $\epsilon = \text{const}$

Boundaries can be grounded conductors (ϕ = 0) or their potential can be specified ($\phi = \phi_0$), e.g., capacitor plates – Dirichlet conditions. We can also consider infinite space, where ϕ = 0 at ∞, but for numerical methods we usually need to make it finite. Then either put ϕ = 0 on the boundaries or estimate ϕ where you put the boundaries and fix ϕ at that value.

Or can have dielectric boundaries with the charge density σ specified, which in the simplest case (e.g., symmetric problem) can give Neumann BC:

$$\epsilon_1 \frac{\partial \phi(\vec{r})}{\partial n}\bigg|_1 - \epsilon_2 \frac{\partial \phi(\vec{r})}{\partial n}\bigg|_2 = -4\pi\sigma(\vec{r}) \Rightarrow \frac{\partial \phi(\vec{r})}{\partial n} = -\frac{2\pi\sigma}{\epsilon} \text{ for a symm. pr.}$$

Heat equation:

$$\frac{\partial T(\vec{r},t)}{\partial t} = \alpha \nabla^2 T(\vec{r},t) + f(\vec{r})$$

$T(\vec{r},t)$ is the temperature

$$\alpha = k/(c_p \rho)$$

$f(\vec{r})$ describes a heat source

Stationary solution:

$$\frac{\partial T(\vec{r},t)}{\partial t} = 0 \;\Rightarrow\; \nabla^2 T(\vec{r}) = -\frac{f(\vec{r})}{\alpha}$$

$T(\vec{r})\big|_\Gamma = T_0(\vec{r})$ – temperature specified at the boundary

$\left.\frac{\partial T(\vec{r})}{\partial n}\right|_\Omega = 0$ – insulating boundary

$\left.\frac{\partial T(\vec{r})}{\partial n}\right|_\Gamma = J(\vec{r})$ – heat flow

Diffusion equation:

$$\frac{\partial c(\vec{r},t)}{\partial t} = D \nabla^2 c(\vec{r},t) + f(\vec{r})$$

$c(\vec{r},t)$ is the particle concentration

$$\nabla^2 c(\vec{r}) = -\frac{f(\vec{r})}{D}$$

$c(\vec{r})\big|_\Gamma = 0$ – absorbing BC

$\left.\frac{\partial c(\vec{r})}{\partial n}\right|_\Gamma = 0$ – reflecting BC

$$\nabla^2 u(\vec{r}) = -f(\vec{r})$$

Finite difference method

Recall 1D: $y'' + g(x)\, y' + k(x)\, y = s(x) \qquad y(a) = C_1;\ y(b) = C_2$

Introduce a mesh $\{x_n\}$, where $x_0=a$, $x_M=b$, $x_n=a+hn$.

$$y_n = y(x_n);\ g_n = g(x_n);\ k_n = k(x_n);\ s_n = s(x_n)$$

Approximate the derivatives by finite differences:

$$y''_n = \frac{y_{n+1} - 2y_n + y_{n-1}}{h^2} + O(h^2);\ y'_n = \frac{y_{n+1} - y_{n-1}}{2h} + O(h^2)$$

$$\frac{y_{n+1} - 2y_n + y_{n-1}}{h^2} + g_n \frac{y_{n+1} - y_{n-1}}{2h} + k_n y_n - s_n + O(h^2) = 0, \quad n = 1, \ldots, M-1$$

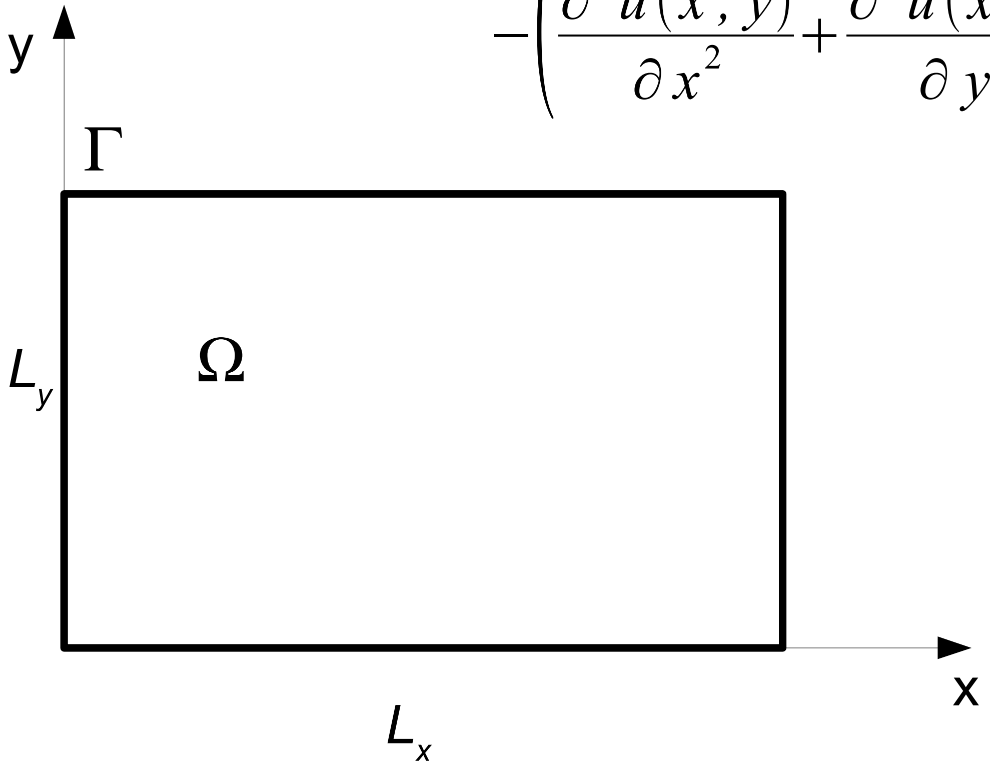
Boundary conditions:

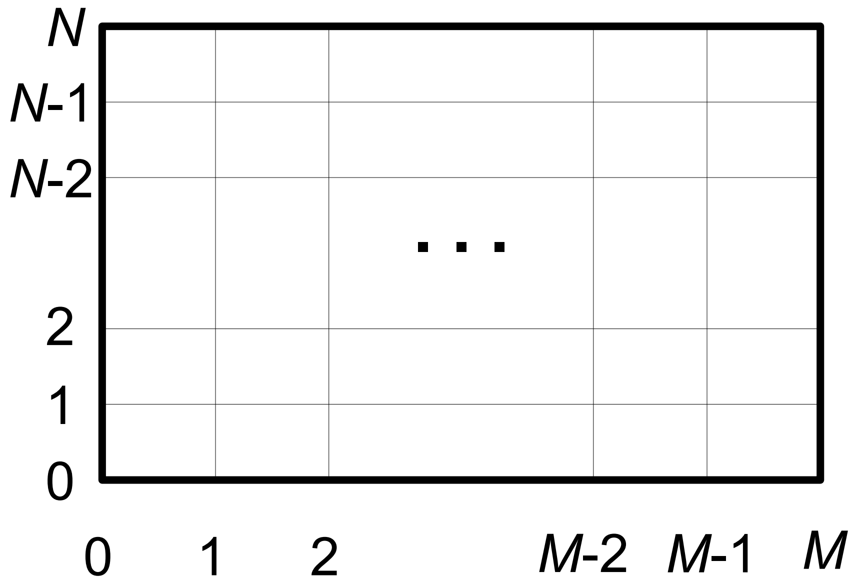
$$y_0 = C_1 \qquad y_M = C_2$$

A system of M+1 linear equations, M+1 unknowns. Tridiagonal. The error is of the same order as the discretization error, i.e., $O(h^2)$.

Straightforward to generalize to higher dimensions. Consider 2D.

$$-\left(\frac{\partial^2 u(x,y)}{\partial x^2}+\frac{\partial^2 u(x,y)}{\partial y^2}\right)=f(x,y) \qquad u|_\Gamma=0$$

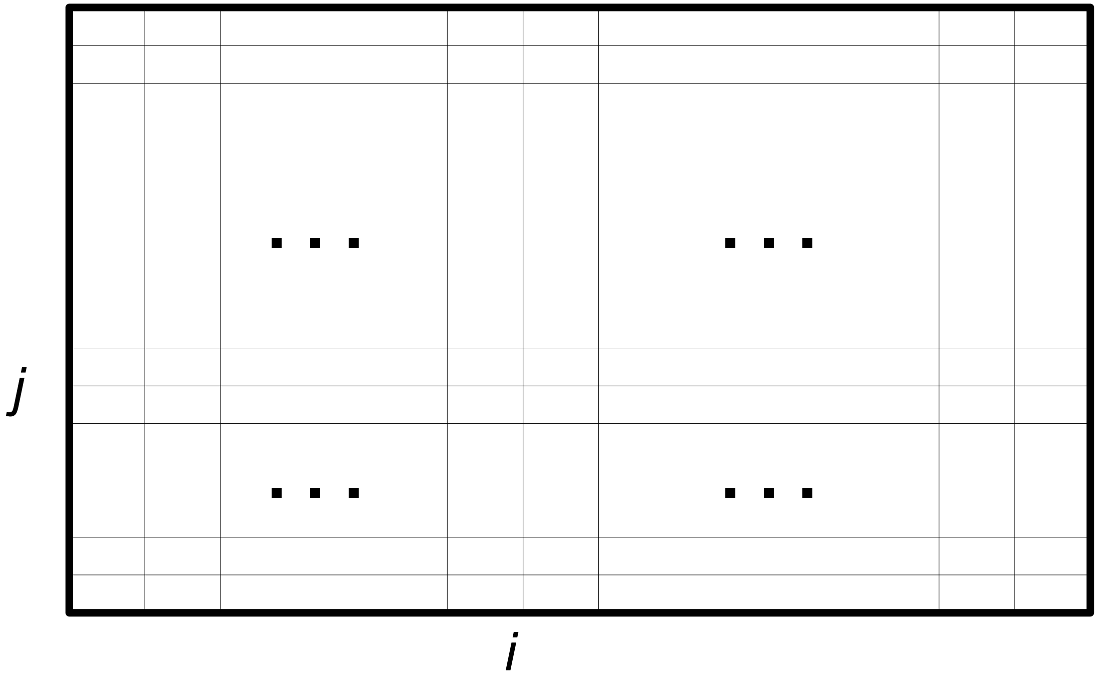


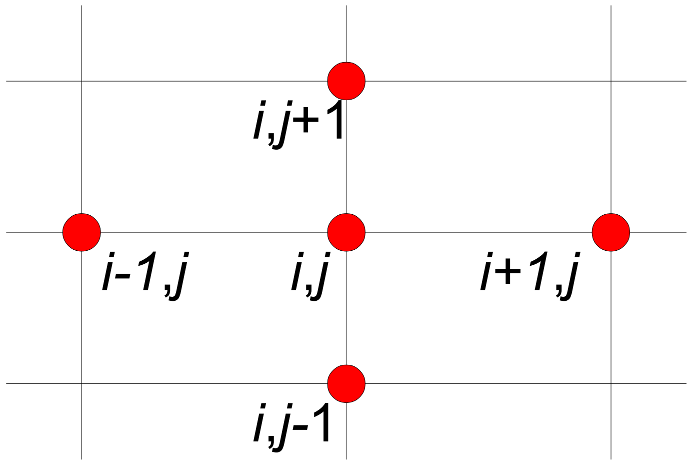


Introduce a rectangular grid $(M+1)\times(N+1)$ $\quad \Delta x=\frac{L_x}{M} \quad \Delta y=\frac{L_y}{N}$

$$-\left(\frac{\partial^2 u(x,y)}{\partial x^2}+\frac{\partial^2 u(x,y)}{\partial y^2}\right)=f(x,y) \qquad u|_\Gamma=0$$

$$x_i=i\,\Delta x \qquad y_j=j\,\Delta y \qquad u(x_i,y_j)\equiv u_{ij} \qquad f(x_i,y_j)=f_{ij}$$





5-point stencil

$$\left.\frac{\partial^2 u(x,y)}{\partial x^2}\right|_{(x_i,y_j)}\approx\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\Delta x^2} \qquad \left.\frac{\partial^2 u(x,y)}{\partial y^2}\right|_{(x_i,y_j)}\approx\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{\Delta y^2}$$

$$u_{0j}=u_{M,j}=u_{i0}=u_{i,N}=0$$

$$-\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\Delta x^2}-\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{\Delta y^2}\approx f_{ij} \qquad i=1,\dots,M-1;\; j=1,\dots,N-1$$

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

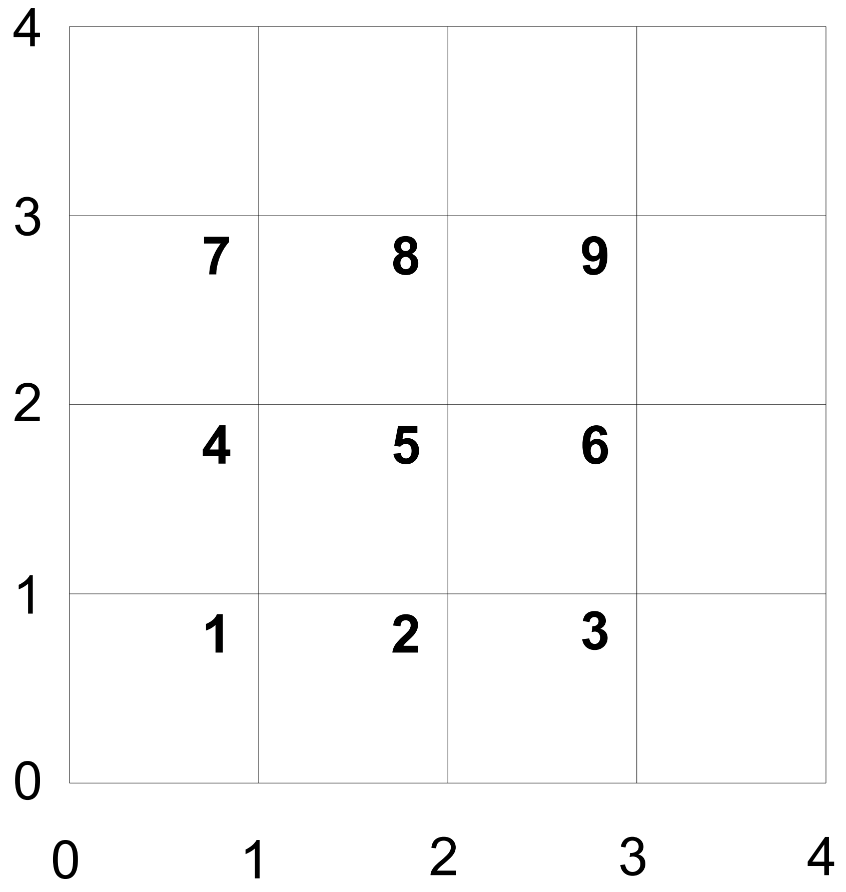
$$i=1,\dots,M-1;\ j=1,\dots,N-1$$

$$u_{0j}=u_{M,j}=u_{i0}=u_{i,N}=0$$

A system of linear equations. (*M*-1)x(*N*-1) equations, as many unknowns.

$$\Delta x=\Delta y=h$$

$$M=N=4$$



For node **1**:

$$\frac{1}{h^2}(-u_{21}+2u_{11}-u_{01}-u_{12}+2u_{11}-u_{10})$$

$$=\frac{1}{h^2}(-u_2+4u_1-u_4)=f_1$$

For node **4**:

$$\frac{1}{h^2}(-u_1-u_5-u_7+4u_4)=f_4$$

For node **5**: $\frac{1}{h^2}(-u_2-u_4-u_6-u_8+4u_5)=f_5$

$$A=\frac{1}{h^2}\left(\begin{array}{ccc|ccc|ccc}
4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\
\hline
-1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\
0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\
\hline
0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4
\end{array}\right)$$

Block-tridiagonal matrix. Block size is *M*-1.

Can also be viewed as a band-diagonal matrix. Band width is 2*M*-1.

$$\vec{v}^T A\vec{v}=\sum_{i=1}^{M-1}\sum_{j=1}^{N-1} a_{ij} v_i v_j=\sum_{i=1}^{M-1}\sum_{j=1}^{N-1}\left[\frac{1}{\Delta x^2}\left(v_{ij}-v_{i-1,j}\right)^2+\frac{1}{\Delta y^2}\left(v_{ij}-v_{i,j-1}\right)^2\right]>0$$

Symmetric positive definite.

$$A\vec{u}=\vec{f}$$

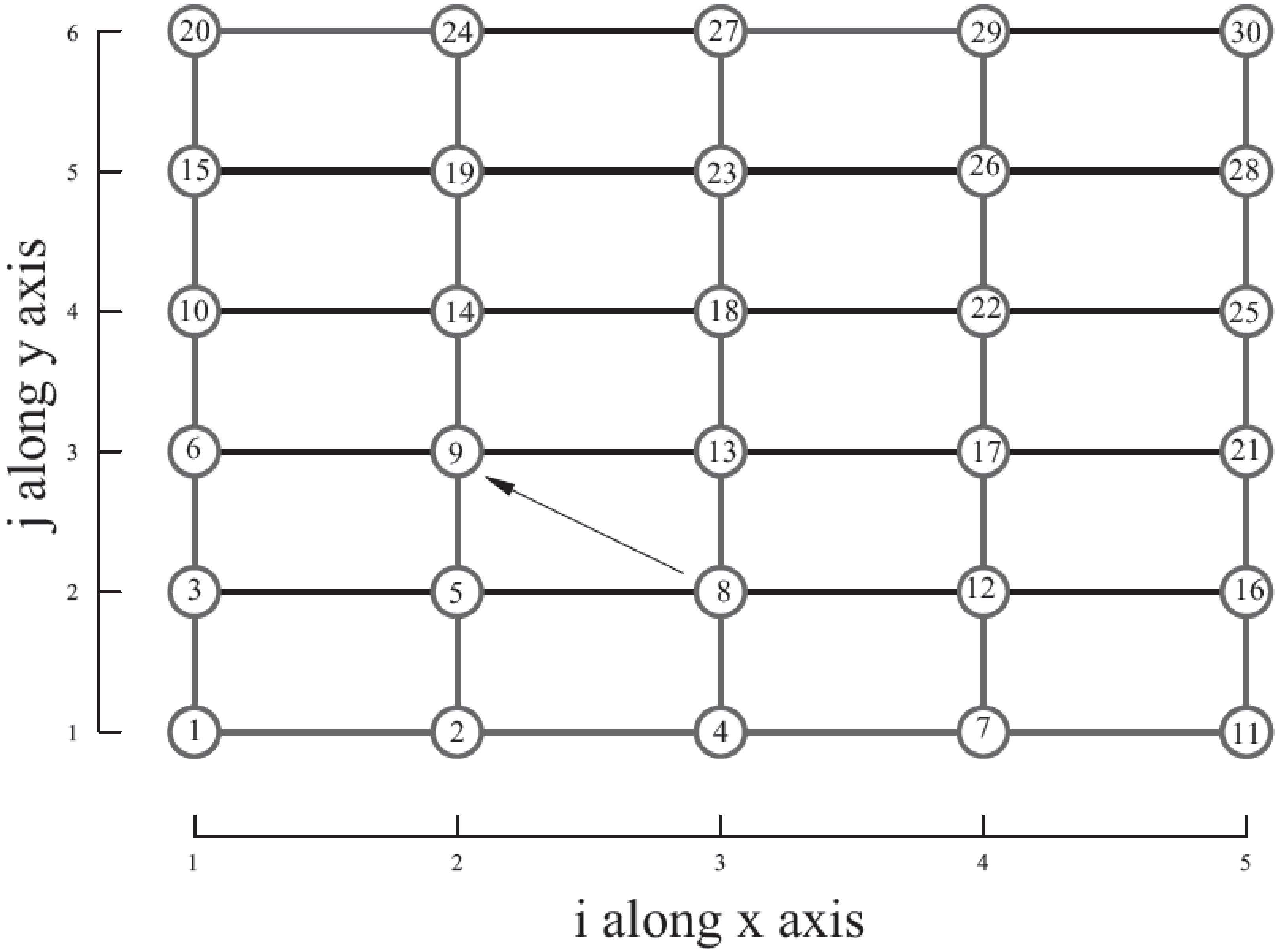
For $M\times M$ grid, the matrix size is $\sim M^2$.

How do we solve the system? Direct methods: Gauss elimination, Cholesky decomposition. For a general matrix, the computational cost would be $\sim\text{size}^3 \sim (M^2)^3 = M^6$. Fortunately, we have a banded matrix, in which case many matrix elements are 0 and remain 0 even during elimination, so the cost is reduced to $\sim \text{size}\times(\text{bandwidth})^2 \sim M^4$. For a rectangular grid, make sure $M<N$.

It may be possible to find better ordering of sites to decrease the amount of “fill” in Gaussian elimination. Diagonal numbering (see next slide), nested dissection. The latter can reduce the cost to $\sim M^3$, which makes it a viable (and more accurate) alternative to iterative methods that I will consider next.

In 3D, however, the matrix size is $\sim M^3$, the band width is $\sim M^2$, so straightforward direct methods give $\sim M^7$ and special sparse methods only get this down to $\sim M^6$, which is extremely costly.

Diagonal numbering



J.R. Hauser, Numerical Methods for Nonlinear Engineering Models, Springer, Dordrecht, 2009

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

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

$$A\vec{x}=\vec{b}:$$

Jacobi: $x_i^{(k+1)}=-\frac{1}{a_{ii}}\sum_{j\neq i}a_{ij}x_j^{(k)}+\frac{1}{a_{ii}}b_i$ (not very useful, but easy to analyze)

$$u_{i,j}^{(k+1)}=\frac{1}{4}\left(u_{i+1,j}^{(k)}+u_{i-1,j}^{(k)}+u_{i,j+1}^{(k)}+u_{i,j-1}^{(k)}\right)+\frac{h^2}{4}f_{ij}$$

At first it seems this should grow indefinitely, but keep in mind the boundary nodes are kept equal to 0. Extension to 1D and 3D is very natural.

A hand-waving argument: this is a discretization of the diffusion equation, or a master equation for particles doing random walks one step per iteration. For $M\times M$ grid, takes ~M^2 steps to travel end to end – ~M^2 iterations to reach equilibrium. Each iteration has cost ~M^2, so total cost ~M^4.

$$\frac{4u_{i,j}-u_{i+1,j}-u_{i-1,j}-u_{i,j+1}-u_{i,j-1}}{h^2}=f_{ij}$$

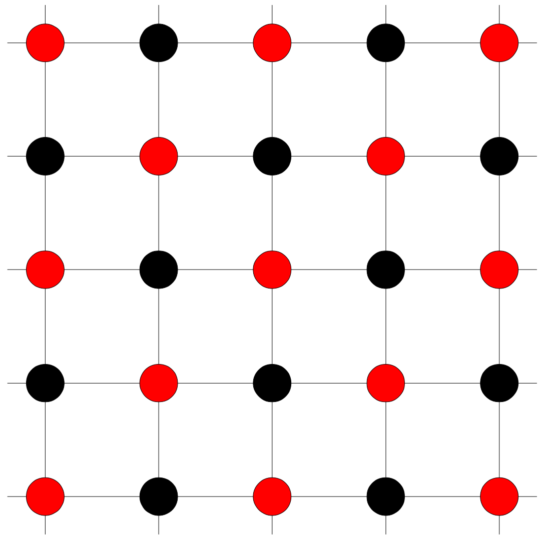
Gauss-Seidel:

$$x_i^{(k+1)}=\frac{1}{a_{ii}}\left(-\sum_{j<i} a_{ij}x_j^{(k+1)}-\sum_{j>i} a_{ij}x_j^{(k)}\right)+\frac{1}{a_{ii}}b_i$$

Depends on the ordering of the sites. If we use the same ordering as before:

$$u_{i,j}^{(k+1)}=\frac{1}{4}\left(u_{i+1,j}^{(k)}+u_{i-1,j}^{(k+1)}+u_{i,j+1}^{(k)}+u_{i,j-1}^{(k+1)}\right)+\frac{h^2}{4}f_{ij}$$

Another popular choice: red-black Gauss-Seidel: update all red sites first, then all black sites.



First for red:

$$u_{i,j}^{(k+1)}=\frac{1}{4}\left(u_{i+1,j}^{(k)}+u_{i-1,j}^{(k)}+u_{i,j+1}^{(k)}+u_{i,j-1}^{(k)}\right)+\frac{h^2}{4}f_{ij}$$

Then for black:

$$u_{i,j}^{(k+1)}=\frac{1}{4}\left(u_{i+1,j}^{(k+1)}+u_{i-1,j}^{(k+1)}+u_{i,j+1}^{(k+1)}+u_{i,j-1}^{(k+1)}\right)+\frac{h^2}{4}f_{ij}$$

Based on the same diffusion argument, it is obvious it should be twice as fast as Jacobi

$$\frac{4u_{i,j}-u_{i+1,j}-u_{i-1,j}-u_{i,j+1}-u_{i,j-1}}{h^2}=f_{ij}$$

Successive over-relaxation (SOR):

$$x_i^{(k+1)}=(1-\omega)x_i^{(k)}+\frac{\omega}{a_{ii}}\left(-\sum_{j<i}a_{ij}x_j^{(k+1)}-\sum_{j>i}a_{ij}x_j^{(k)}+b_i\right)$$

For red:

$$u_{i,j}^{(k+1)}=(1-\omega)u_{i,j}^{(k)}+\frac{\omega}{4}\left(u_{i+1,j}^{(k)}+u_{i-1,j}^{(k)}+u_{i,j+1}^{(k)}+u_{i,j-1}^{(k)}+h^2 f_{ij}\right)$$

Then for black:

$$u_{i,j}^{(k+1)}=(1-\omega)u_{i,j}^{(k)}+\frac{\omega}{4}\left(u_{i+1,j}^{(k+1)}+u_{i-1,j}^{(k+1)}+u_{i,j+1}^{(k+1)}+u_{i,j-1}^{(k+1)}+h^2 f_{ij}\right)$$

Optimal ω can be obtained from the spectral radius of the Jacobi iteration λ_J

$$\omega_{\text{opt}}=\frac{2}{1+\sqrt{1-\lambda_J^2}}.$$ In our case, $$\lambda_J=\frac{\cos(\pi/M)+\beta^2\cos(\pi/N)}{1+\beta^2};\quad \beta=\Delta x/\Delta y$$

There are ways to estimate ω when it is not known analytically.

Symmetric SOR (switch red $\leftrightarrow$ black) supposedly leads to an improvement.

Line SOR: solve the system for a line of sites directly using the values for adjacent lines from the previous iteration or from the current iteration where available. Over-relax.

$$A=\frac{1}{h^2}\left(\begin{array}{ccc|ccc|ccc}
4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\
\hline
-1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\
0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\
\hline
0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4
\end{array}\right)$$

Speeds up convergence by a factor $\sqrt{2}$.

Alternating between lines and columns, gives alternating direction implicit method. Optimal method includes cycling between different values of ω.

Conjugate gradient method can be used, but preconditioning is desirable.

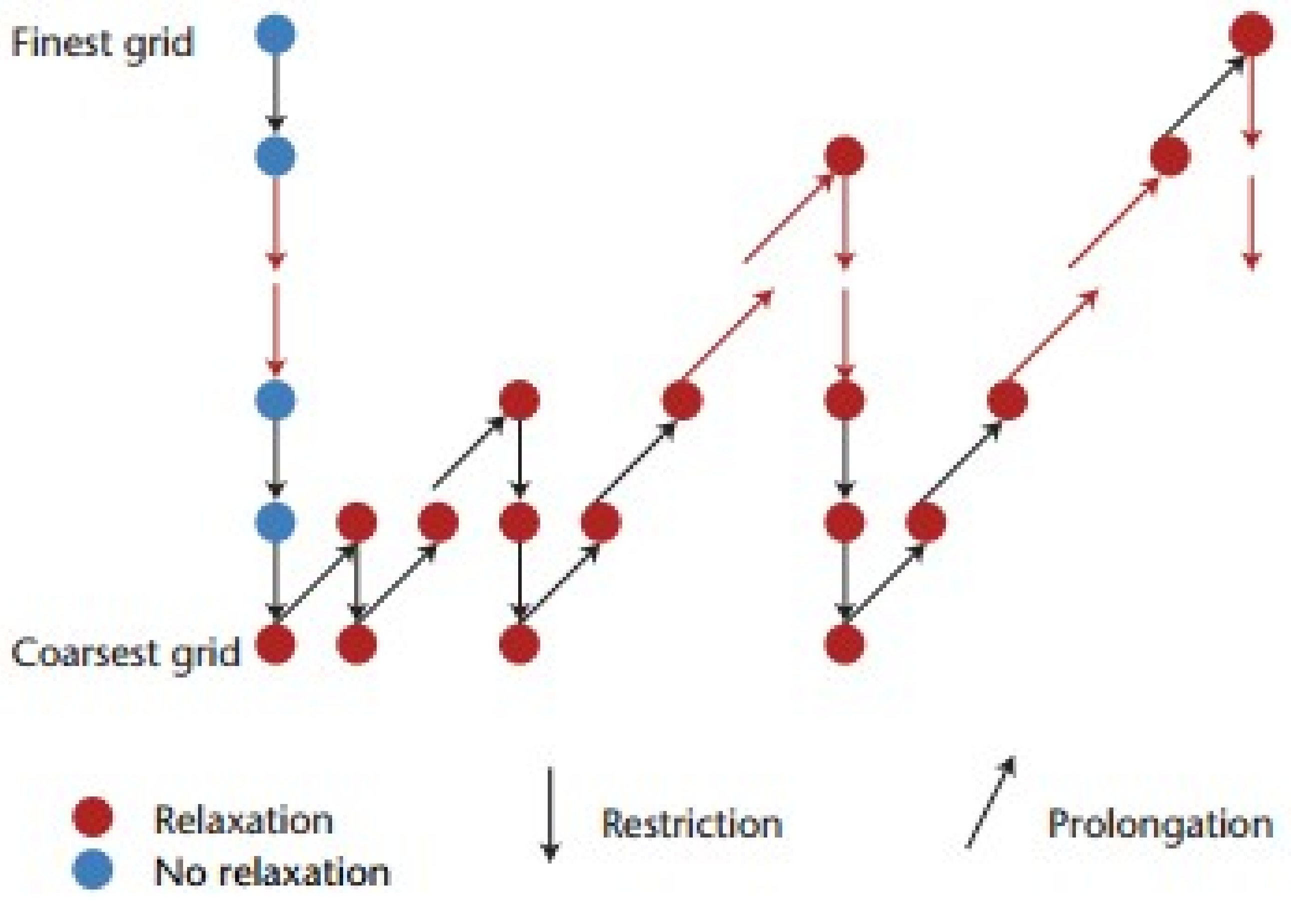
Multigrid approach

In many iterative methods, high-frequency components relax faster than low-frequency components. Such methods are called smoothers. Not true for Jacobi, true for sufficiently underrelaxed Jacobi, Gauss-Seidel, SOR. Coarser discretizations also relax faster.

Idea: alternate relaxations on coarse grids that would relax large features rapidly and relaxations on fine grids that would relax high-frequency features rapidly.

Need procedures that transfer from fine to coarse grid (restriction) and vice versa (prolongation). Restriction usually done by just taking the value at the coarse grid points or some weighted average. Prolongation can be done by linear interpolation.

Full multigrid:



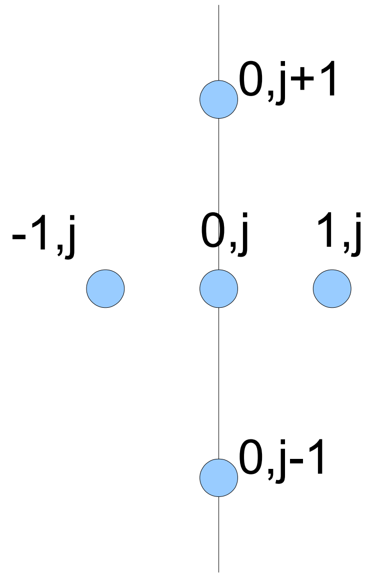
Each “V-cycle” can be repeated several times. Achieves linear cost in the number of grid points [i.e., O(M^2) in 2D].

Variations of the problem

Neumann boundary condition

$$\left.\frac{\partial u(\vec{r})}{\partial n}\right|_{\Gamma} = g(\vec{r})$$

Just as in 1D, introduce a “ghost” site behind the boundary.



$$-\frac{u_{1,j}-2u_{0,j}+u_{-1,j}}{\Delta x^2}-\frac{u_{0,j+1}-2u_{0,j}+u_{0,j-1}}{\Delta y^2}=f_{0,j}$$

$$\frac{u_{1,j}-u_{-1,j}}{2\Delta x}=g_{0,j} \;\Rightarrow\; u_{-1,j}=u_{1,j}-2g_{0,j}\Delta x$$

$$-2\frac{u_{1,j}-u_{0,j}-g_{0,j}\Delta x}{\Delta x^2}-\frac{u_{0,j+1}-2u_{0,j}+u_{0,j-1}}{\Delta y^2}=f_{0,j}$$

Periodic boundary conditions

$$u_{M,j}=u_{0,j} \text{ etc.}$$

Other terms in the equation

$$D\nabla^2 u - \vec{v}\cdot\nabla u - Cu = -f$$

$$\left.\frac{\partial u}{\partial x}\right|_{(x_i,y_j)} \approx \frac{u_{i+1,j}-u_{i-1,j}}{2\Delta x} \qquad u(x_i, x_j) \rightarrow u_{i,j}$$

But, similar to our discussion for ODEs, $D\nabla^2 u - v\dfrac{\partial u}{\partial x} = 0$

$$D\frac{u_{i+1,j}+u_{i-1,j}+u_{i,j+1}+u_{i,j-1}-4u_{i,j}}{h^2} - v\frac{u_{i+1,j}-u_{i-1,j}}{2h} = 0$$

Péclet number $\mathrm{Pe} = \dfrac{vh}{2D}$. If Pe>1, may no longer be positive definite.

Then use $\dfrac{u_{i,j}-u_{i-1,j}}{\Delta x}$.

Eigenvalue problem

$$-\nabla^2 u = \lambda u$$

$$-\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\Delta x^2} - \frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{\Delta y^2} = \lambda u_{ij}$$

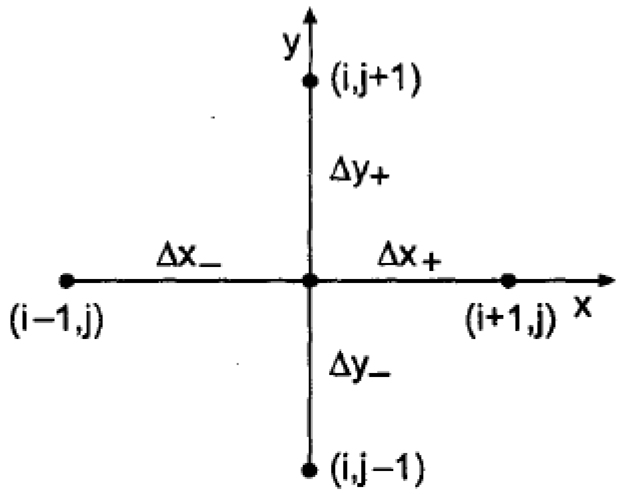
Algebraic eigenvalue problem

Non-rectangular domains

Straightforward, if the boundaries are still along the grid lines (although there may be issues with corners).

If that is not the case, finite volume and especially finite element may be better approaches. But finite differences can still handle it.

Crudest: approximate the boundary with straight lines along the grid.



J.D. Hoffman, Numerical Methods for Engineers and Scientists, Marcel Dekker, New York, 2001.

Nonuniform finite-difference approximations

$$u_{i+1,j} \approx u_{i,j} + u_x\big|_{i,j} \Delta x_+ + \frac{1}{2} u_{xx}\big|_{i,j} \Delta x_+^2$$

$$u_{i-1,j} \approx u_{i,j} - u_x\big|_{i,j} \Delta x_- + \frac{1}{2} u_{xx}\big|_{i,j} \Delta x_-^2$$

$$u_{xx}\big|_{i,j} \approx 2 \frac{\Delta x_+ u_{i-1,j} - (\Delta x_- + \Delta x_+) u_{i,j} + \Delta x_- u_{i+1,j}}{\Delta x_- \Delta x_+^2 + \Delta x_+ \Delta x_-^2}$$

An even better alternative: coordinate transformation

$(x, y) \rightarrow (\xi, \eta)$ such that the boundaries correspond to $\xi = \text{const}$,

$\eta = \text{const}$, or both.

$$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial u}{\partial \eta}\frac{\partial \eta}{\partial x} \qquad \frac{\partial u}{\partial y} = \frac{\partial u}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial u}{\partial \eta}\frac{\partial \eta}{\partial y}$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial \xi^2}\left(\frac{\partial \xi}{\partial x}\right)^2 + \frac{\partial u}{\partial \xi}\frac{\partial^2 \xi}{\partial x^2} + \frac{\partial^2 u}{\partial \eta^2}\left(\frac{\partial \eta}{\partial x}\right)^2 + \frac{\partial u}{\partial \eta}\frac{\partial^2 \eta}{\partial x^2} + 2\frac{\partial^2 u}{\partial \xi \partial \eta}\frac{\partial \xi}{\partial x}\frac{\partial \eta}{\partial x}$$

$$\frac{\partial^2 u}{\partial y^2} = \frac{\partial^2 u}{\partial \xi^2}\left(\frac{\partial \xi}{\partial y}\right)^2 + \frac{\partial u}{\partial \xi}\frac{\partial^2 \xi}{\partial y^2} + \frac{\partial^2 u}{\partial \eta^2}\left(\frac{\partial \eta}{\partial y}\right)^2 + \frac{\partial u}{\partial \eta}\frac{\partial^2 \eta}{\partial y^2} + 2\frac{\partial^2 u}{\partial \xi \partial \eta}\frac{\partial \xi}{\partial y}\frac{\partial \eta}{\partial y}$$

$$u_{xx} + u_{yy} = a(\xi, \eta) u_{\xi\xi} + b(\xi, \eta) u_{\xi\eta} + c(\xi, \eta) u_{\eta\eta} + d(\xi, \eta) u_{\xi} + e(\xi, \eta) u_{\eta}$$

Nonlinear equations

1. Iteration. Discretize, obtain a set of nonlinear finite-difference equations.

$$\boldsymbol{A}\vec{u}=f(\vec{u})$$
$$\boldsymbol{A}\vec{u}^{(k+1)}=f(\vec{u}^{(k)})$$

2. Newton's method.

$$u(x,y)=U(x,y)+\delta u(x,y)$$

Linearize with respect to $\delta u(x,y)$, solve the linear system. Update

$$U^{(k+1)}(x,y)=U^{(k)}(x,y)+\delta u^{(k)}(x,y)$$

Both methods involve solving a linear system at each iteration. If this itself is done iteratively, no need to solve to convergence at early stages.

Boundary element method

$$\nabla^2 \phi = 0$$

$$\nabla^2 G(\vec{r}, \vec{r}\,') = -\delta(\vec{r} - \vec{r}\,')$$

$$\nabla \cdot [G(\vec{r} - \vec{r}\,') \nabla(\phi(\vec{r})) - \phi(\vec{r}) \nabla(G(\vec{r} - \vec{r}\,'))] = -\phi(\vec{r}) \nabla^2 (G(\vec{r} - \vec{r}\,'))$$

$$\oint_{\partial V} dA \left(G(\vec{r} - \vec{r}\,') \frac{\partial}{\partial n} (\phi(\vec{r})) - \phi(\vec{r}) \frac{\partial}{\partial n} (G(\vec{r} - \vec{r}\,')) \right)$$

$$= -\int_V dV (\phi(\vec{r}) \nabla^2 (G(\vec{r} - \vec{r}\,'))) = \phi(\vec{r}\,')$$