[4] Accuracy, the 9-pt. Laplacian.

→ For the 5-pt. Laplacian, we easily see that the method is 2nd-order in both x & y directions.

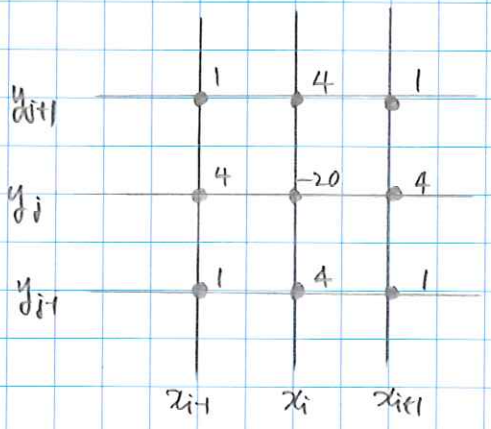
→ the local truncation error E_q is given as

$$E_q = \frac{1}{h^2}\left[u(x_{i-1}, y_j) + u(x_{i+1}, y_j) + u(x_i, y_{j-1}) + u(x_i, y_{j+1}) - 4u(x_i, y_j)\right] - f(x_i, y_j)$$

$$= \frac{h^2}{12}\left(u_{xxxx} + u_{yyyy}\right) + O(h^4).$$

① 2nd order, in both x & y ($\Delta x = \Delta y = h$)

→ Compared to this, one can write the 9-pt Laplacian on the 9-pt. stencil:



$$\nabla_9^2 U_{ij} = \frac{1}{6h^2}\left[U_{i-1,j+1} + 4U_{i,j+1} + U_{i+1,j+1} + 4U_{i-1,j} - 20U_{ij} + 4U_{i+1,j} + U_{i-1,j-1} + 4U_{i,j-1} + U_{i+1,j-1}\right]$$

→ One can show that the relationship between the 5-pt and the 9-pt Laplacians is:

$$\nabla_9^2 u(x_i, y_j) = \nabla^2 u(x_i, y_j) + \underbrace{\frac{h^2}{12}\left(u_{xxxx} + 2u_{xxyy} + u_{yyyy}\right)}_{①} + O(h^4)$$

(!) The local truncation error for $\nabla_9^2 u$ is still $O(h^2)$.

→ The 9-pt. Laplacian is 2nd order in both x & y.

→ Although there seems no gain in $\nabla_9^2 u$ in terms of soln accuracy over the 5-pt. Laplacian $\nabla^2 u$, there is one very nice advantage when calculating the leading error term due to ①.

→ Note that, assuming $\nabla^2 u = f$, f: known

$$① = \frac{h^2}{12}\nabla^2(\nabla^2 u)$$

$$= \frac{h^2}{12}\nabla^2 f$$

→ Hence the leading error term ① can be easily computed <u>without</u> knowing the soln u, but can be directly computed from the known fkn f.

→ In particular, if $f = 0$, then ① $= 0$, yielding $\nabla_9^2 u$ a fourth-order method.

→ In general, if $f \neq 0$, one can still obtain a fourth-order method of the form

$$\nabla_9^2 v_{ij} = f_{ij}, \quad \cdots ②$$

where, for $f(x,y)$: arbitrary smooth ftns, we define

$$f_{ij} = f(x_i, y_j) + \frac{h^2}{12} \nabla^2 f(x_i, y_j) . \quad \cdots ③$$

→ Then we get, from ②,

$$\nabla_9^2 U_{ij} = \nabla^2 U_{ij} + \frac{h^2}{12} \nabla^2 f(x_i, y_j) + O(h^4)$$

$$\| $$

$$f_{ij} = f(x_i, y_j) + \frac{h^2}{12} \nabla^2 f(x_i, y_j)$$

∴ The $O(h^2)$ terms cancel out, giving a fourth-order method.

→ This is to be considered as deliberately introducing an $O(h^2)$ error to the right hand side of the eqn, so that the $O(h^2)$ error cancels out perfectly.

→ In case we do NOT have an analytic form of $f(x,y)$, but only know $f(x_i, y_j)$ at the grid pts, we use

$$f_{ij} = f(x_i, y_j) + \frac{h^2}{12} \nabla_5^2 f(x_i, y_j)$$

instead of ③, in order to obtain fourth-order.

5 Two ways of solving the linear system

→ The large linear systems arise from discretizing elliptic eqns. Two approaches are available:

(1) Direct solvers:

→ provide the exact soln x, modulo rounding errors

→ (ex) Gaussian elimination, LU decomposition, Gauss-Jordan elimination, Cholesky decomposition for SPD, Crout's method for LU, etc.

→ The matrix needs to be stored, at least partially.

(2) Indirect solvers:

→ start with a first approximation (guess) $x^{(0)}$, & compute iteratively a sequence of (hopefully increasingly better) approximations $x^{(k)}$, without ever reaching the exact soln x.

→ (ex) Jacobi, Gauss-Seidel, descent method, Conjugate gradient (CG) (→ for SPD), incomplete Cholesky, incomplete LU (ILU), Arnoldi process & GMRES (generalized minimum residual), Newton-Krylov methods (nonlinear problems), multigrid methods, etc.

→ The matrix is never stored, and at most need to store nonzero elements.

Rmk Operation count in Gaussian elimination (GE) method:

→ Given a general $N \times N$ dense matrix, GE requires $O(N^3)$ operations.

→ Applying a general GE can be very expensive!

(e.g.) 3D Poisson problem on $100 \times 100 \times 100$ grid

→ $N = 100^3 = 10^6$, ☹ $N^3 = 10^{18}$.

→ Considering a typical laptop (~gigaflop) to run the problem, assuming 10 gigaflops ($= 10 \times 10^9 = 10^{10}$ floating pt. operation/sec) it takes about

$$10^{18}/10^{10} = 10^8 \text{ sec} > 3 \text{ years}.$$

→ One therefore needs faster approaches, especially for sparse matrices, using efficient algorithms based on, preferably, indirect solvers.

1 year $\hat{=}$ 3.15×10^7 sec.

deca	10^1	10^{-1}	deci
hecto	10^2	10^{-2}	centi
kilo	10^3	10^{-3}	milli
mega	10^6	10^{-6}	micro
giga	10^9	10^{-9}	nano
tera	10^{12}	10^{-12}	pico
peta	10^{15}	10^{-15}	femto
exa	10^{18}	10^{-18}	atto