Mat1062: Introductory Numerical Methods for PDE

Mary Pugh

January 29, 2009

1 Ownership

These notes are the joint property of Rob Almgren and Mary Pugh.

2 Diffusion equation in higher dimensions

Now let's talk about the diffusion equation in two and three dimensions:

$$u_t = D\,\Delta u$$

where $\Delta u = \nabla^2 u = \mathrm{div}(\nabla u) = u_{xx} + u_{yy}$ or $u_{xx} + u_{yy} + u_{zz}$. It is to be solved on some domain Ω, with initial data $u_0(x)$ and some mixture of Dirichlet or Neumann boundary conditions on the boundary $\partial\Omega$.

3 Properties of Equation

Conservation For a fixed domain V,

$$\frac{d}{dt}\int_V u(x,t)\,dV = \int_V u_t(x,t)\,dV = \int_V D\ \mathrm{div}(\nabla u)\,dV = \int_{\partial V} D\,\frac{\partial u}{\partial n}\,dA.$$

where dV is volume element inside V, and dA is surface area element on the boundary ∂V. As before, nothing is created or destroyed inside; changes happen only across the boundary. It tells you that if D is not constant, then the equation should (barring some special effects) be written *not* as $u_t = D(x,t,u)\Delta u$ but as $u_t = \mathrm{div}(D(x,t,u)\nabla u)$, in order to preserve conservation, and that this structure should be preserved in the discretization.

Maximum principle At a local maximum of u, all second derivatives are nonpositive, and hence $\Delta u \le 0$, so $u_t \le 0$ and the maximum cannot increase. Thus (unless forced by the boundary conditions) the maximum and minimum values of u at later times are bounded by the maximum and minimum of the initial data. In particular, if the solution is initially nonnegative it cannot become negative at later times, and this property should be preserved by the discretization. Since the maximum principle also applies to derivatives of u, the solution gets smoother as time evolves.

And we again have the same two special solutions.

Green's function In d dimensions (with D constant), we look for a radially symmetric similarity solution of the form

$$u(x,t) \;=\; \frac{1}{(Dt)^{d/2}}\, U\!\left(\frac{r}{\sqrt{Dt}}\right)$$

in which the prefactor must have this form by conservation: the length scale of $U(r/\sqrt{Dt})$ is $\sqrt{Dt}$, so mass spreads over a volume $(Dt)^{d/2}$ in d dimensions.

Then we write the PDE in d-dimensional spherical coordinates as

$$\frac{\partial u}{\partial t} \;=\; \frac{D}{r^{d-1}}\frac{\partial}{\partial r}\left(r^{d-1}\frac{\partial u}{\partial r}\right) \;=\; D\left(\frac{\partial^2 u}{\partial r^2} \;+\; \frac{d-1}{r}\,\frac{\partial u}{\partial r}\right) \tag{1}$$

which gives the ODE in terms of the similarity variable $\rho = r/\sqrt{Dt}$

$$U''(\rho) \;+\; \left(\frac{d-1}{\rho} + \frac{\rho}{2}\right) U'(\rho) \;+\; \frac{d}{2}\, U(\rho) \;=\; 0.$$

By inspection, we see that a solution is

$$U(\rho) \;=\; C\, e^{-\rho^2/4}$$

and we determine C by integration. We require

$$\begin{aligned} 1 \;&=\; \int_{\mathbb{R}^d} u(r,t)\, dV \;=\; \int_0^\infty u(r,t)\, dV(r) \;=\; \int_0^\infty U(\rho)\, dV(\rho) \\ &=\; C\int_0^\infty e^{-\rho^2/4}\, dV(\rho) \;=\; C\left(\int_{-\infty}^{\infty} e^{-x^2/4}\, dx\right)^d \;=\; C\big(2\sqrt{\pi}\big)^d \end{aligned}$$

where $dV = dx_1 \cdots dx_d$ and $dV(r) = d\omega_d r^{d-1}$ where ω_d is the volume of the d-dimensional unit sphere. Hence $C = 1/(\sqrt{4\pi})^d$ and the final Green's function is

$$G(x,t) \;=\; \frac{1}{(4\pi Dt)^{d/2}}\, e^{-r^2/4Dt}$$

Fourier modes The other class of special solutions are the Fourier modes, with spatial structure $\sin(\xi_1 x_1)\cdots\sin(\xi_d x_d)$. Plugging in the PDE gives

$$u(x,t) \;=\; e^{-\sigma t}\sin(\xi_1 x_1)\cdots\sin(\xi_d x_d), \qquad \sigma = D\xi^2 = D\big(\xi_1^2 + \cdots + \xi_d^2\big).$$

The same function could be written in terms of cosines. In fact, to see the spatial structure it is much easier to write it as complex exponentials:

$$u(x,t) \;=\; \exp\big(-\sigma t \,+\, i(\xi_1 x_1 + \cdots + \xi_d x_d)\big) \;=\; \exp\big(-\sigma t \,+\, i\xi\cdot x\big),$$

where $\xi = (\xi_1, \ldots, \xi_d)$ is the *wave vector*. These are plane waves with normal vector $\xi/|\xi|$, and of wavelength $2\pi/|\xi|$. Except for the rotational degree of freedom, the dispersion relation is the same as in one dimension.

4 Discretization

The simplest case is a rectangular box. To simplify the notation, let's write everything in just two dimensions. The box is $[0, L_x] \times [0, L_y]$, with either Dirichlet or Neumann boundary conditions on each wall or segment of wall independently.

4.1 Space discretization

We pick grid sizes N_x, N_y, giving spacings $h_x = L_x/N_x$ and $h_y = L_y/N_y$. We discretize a smooth function $u(x,y,t)$ in space by the $(N_x+1)(N_y+1)$ grid values

$$u_{i,j}(t) \;\approx u\big(\,ih_x,\, jh_y,\, t\big), \qquad i = 0, \ldots, N_x,\; j = 0, \ldots, N_y.$$

We suppose the boundary values are stored, even though this is not strictly necessary for Dirichlet boundaries. We will reserve k for the time step, so in three dimensions we would either have to introduce indices i_1, i_2, i_3 or change k to τ and use i, j, k.

Straightforward expansion in each dimension separately shows that

$$\begin{aligned}
\frac{1}{h_x^2}\big(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}\big) &= u_{xx} + \frac{1}{12}\,h_x^2\,u_{xxxx} + \cdots \\
\frac{1}{h_y^2}\big(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}\big) &= u_{yy} + \frac{1}{12}\,h_y^2\,u_{yyyy} + \cdots
\end{aligned}$$

Simplifying the most common case of a square grid on which $h_x = h_y$ (obtained by choosing $N_x/N_y = L_x/L_y$), we have the 5-point approximation

$$\begin{aligned}
(Lu)_{i,j} &= \frac{1}{h^2}\big(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}\big) \\
&\sim \Delta u + \frac{1}{12}\,h^2\,(u_{xxxx} + u_{yyyy}) + \cdots
\end{aligned}$$

We thus get the system of ODEs

$$\frac{d}{dt}\,u_{i,j}(t) = \frac{D}{h^2}\big(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}\big).$$

In three dimensions the obvious extension gives the 7-point Laplacian.

The five-point stencil used the nearest neighbors of $u_{i,j}$ — it used $u_{i,j+1}$, $u_{i,j-1}$, $u_{i-1,j}$, and $u_{i+1,j}$. A more appealing formula can be constructed by using the next-nearest neighbors as well:

$$\begin{aligned}
(Lu)_{i,j} &= \frac{1}{6h^2}\,(u_{i+1,j+1} + u_{i-1,j+1} + u_{i-1,j-1} + u_{i+1,j+1}) \\
&\qquad + \frac{2}{3h^2}\,(u_{i,j+1} + u_{i,j-1} + u_{i-1,j} + u_{i+1,j}) - \frac{10}{3h^2}u_{i,j} \\
&\sim \Delta u + \frac{1}{12}\,h^2\,\Delta\Delta u + \cdots
\end{aligned}$$

Note that even though though the truncation error is of precisely the same size (h^2) it has the form $\Delta\Delta u$ which is more isotropic, meaning independent of grid orientation—in some problems this makes a difference. It is consistent with the interpretation of the Laplacian as average value around a small circle minus value at the center: including the corner points rather than just the edge points gives a fuller approximation. Stability properties are slightly improved as well. This can be generalized to three dimensions, resulting in a 27-point stencil.

4.2 Time discretization

We do time discretization exactly as before. The fully explicit method is

$$u_{i,j}^{n+1} \;=\; u_{i,j}^{n} \;+\; \lambda\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}\right), \qquad \lambda \;=\; \frac{Dk}{h^2}.$$

An implicit discretization with parameter $\frac{1}{2} \le \theta \le 1$ requires solving the linear system

$$\left(I - \theta DkL\right) u^{n+1} \;=\; \left(I + (1-\theta)DkL\right) u^{n}$$

where L denotes the discrete Laplacian operator above, and u^n, u^{n+1} denote the entire vector of grid values, of length $(N_x+1)(N_y+1)$ except for boundary values corresponding to Dirichlet boundaries. (In the latter case, inhomogeneous terms should also be added on the right side, just as in 1-D.)

In one dimension, the matrix on the left is tridiagonal, since it couples each point only to its nearest neighbors. Hence it is easily solved in only a little more time than needed for the explicit scheme.

In two or three dimensions, the structure of the matrix depends on how the points of the two-dimensional grid are ordered into a one-dimensional vector. The most natural way is to do it row-by-row:

$$u \;=\; \begin{pmatrix} u_{0,0} & \dots & u_{N_x,0} & u_{0,1} & \dots & u_{N_x,1} & \dots & u_{0,N_y} & \dots & u_{N_x,N_y} \end{pmatrix}^{T}.$$

(The ^{T} denotes transpose, and indicates that u is to be a column vector so it can be multiplied on the left by A.) Then, since L couples nearest neighbors in the y-direction as well as in the x-direction, the matrix A has *five* nonzero diagonals: the main diagonal representing the influence of point (i,j) on itself, the diagonals just above and just below corresponding to the neighbors $(i \pm 1, j)$, and two diagonals a distance $N_y + 1$ away from the main diagonal, corresponding to the neighbors $(i, j \pm 1)$. In fact, the matrix L may be represented as

$$L \;=\; \frac{1}{h^2}\begin{pmatrix}
\ddots & & & & \ddots & \ddots & \ddots & & & & \ddots & & & & \\
 & 1 & & & & 1 & -4 & 1 & & & & 1 & & & \\
 & & 1 & & & & 1 & -4 & 1 & & & & 1 & & \\
 & & & 1 & & & & 1 & -4 & 1 & & & & 1 & \\
 & & & & \ddots & & & & \ddots & \ddots & \ddots & & & & \ddots
\end{pmatrix}.$$

and A has the same structure.

In three dimensions, the situation is even worse: there are two diagonals a distance N_x away, as here, and in addition two more a distance $N_x N_y$ corresponding to the neighbors above and below. Thus in two and three dimensions, inverting the matrix for the linear system is a highly nontrivial problem. There is no way to rearrange the point ordering to make the situation substantially better.

Postponing discussion of the linear algebra, let us analyse consistency and stability. In fact, we analysed the truncation error above: the total error is $\mathcal{O}(h^2, k)$ if $\theta \neq \frac{1}{2}$, and $\mathcal{O}(h^2, k^2)$ if $\theta = \frac{1}{2}$, just as in one dimension.

4.3 Stability

We now do the von Neumann analysis of the stability. As before, we ignore boundaries and consider how the time-stepping scheme would perform given initial data on $h\mathbb{Z} \times h\mathbb{Z}$. (How to modify things for $h_x\mathbb{Z} \times h_y\mathbb{Z}$ follows naturally.)

In one dimension, given $\{u_m\}$ defined on $h\mathbb{Z}$ we used it to define $\widehat{u}(\xi)$ where $\xi \in [-\pi/h, \pi/h]$. In two dimensions, given $\{u_{i,j}\}$ defined on $h\mathbb{Z} \times h\mathbb{Z}$, we define $\widehat{u}(\xi, \eta)$ for $\xi, \eta \in [-\pi/h, \pi/h]$. Proceeding exactly as before, the discrete scheme

$$u_{i,j}^{n+1} \;=\; u_{i,j}^{n} \;+\; \lambda\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}\right),$$

transforms into

$$\begin{aligned}\widehat{u}^{n+1}(\xi,\eta) &= (1-4\lambda)\widehat{u}^{n}(\xi,\eta) + \lambda e^{h\xi\sqrt{-1}}\widehat{u}^{n}(\xi,\eta) + \lambda e^{-h\xi\sqrt{-1}}\widehat{u}^{n}(\xi,\eta) \\ &\quad + \lambda e^{h\eta\sqrt{-1}}\widehat{u}^{n}(\xi,\eta) + \lambda e^{-h\eta\sqrt{-1}}\widehat{u}^{n}(\xi,\eta) \\ &= (1 - 4\lambda + 2\cos(h\xi) + 2\cos(h\eta))\;\widehat{u}^{n}(\xi,\eta)\end{aligned}$$

The scheme is stable if the prefactor $1 - 4\lambda + 2\cos(h\xi) + 2\cos(h\eta)$ has magnitude less than or equal to 1 for all $\xi, \eta \in [-\pi/h, \pi/h]$. As before, we can study this prefactor using calculus and we find that it's most negative values occur when $\xi = \pm\pi/h$ and $\eta = \pm\pi/h$. For these values to be greater than or equal to -1 we find that $\lambda \leq 1/4$.

Here is a fully equivalent, somewhat less cumbersome, way of doing the same stability analysis. We look for special solutions having the form

$$u_{i,j}^{n} \;=\; \omega_1^{i}\,\omega_2^{j}\,\eta^{n},$$

where ω_1 and ω_2 are two complex numbers of unit magnitude (superscripts on ω_1, ω_2, and η indicate exponentiation). The discretization formula gives us η in terms of ω_1, ω_2; the scheme is stable if $|\eta| \le 1$ for all such ω_1, ω_2.

We begin by calculating the effect of the discrete Laplacian operator applied to such a system.

$$\begin{aligned}
(Lu)^n_{i,j} &= \frac{1}{h^2}\left(\frac{1}{\omega_1} + \omega_1 + \frac{1}{\omega_2} + \omega_2 - 4\right)\omega_1^i\,\omega_2^j\,\eta^n \\
&= \frac{1}{h^2}\Big[\,2\,\mathrm{Re}\,\omega_1 \;+\; 2\,\mathrm{Re}\,\omega_2 \;-4\Big]\omega_1^i\,\omega_2^j\,\eta^n \\
&= -\frac{2}{h^2}\Big[\,\big(1-\mathrm{Re}\,\omega_1\big) \;+\; \big(1-\mathrm{Re}\,\omega_2\big)\,\Big]\omega_1^i\,\omega_2^j\,\eta^n
\end{aligned}$$

We now use that $\omega_1^i\omega_2^j\eta^n$ is a solution of the discrete scheme:

$$\omega_1^i\omega_2^j\eta^{n+1} = \omega_1^i\omega_2^j\eta^n + Dk(Lu)^n_{i,j}$$

which then implies

$$\eta = 1 - 2\lambda\Big[\,\big(1-\mathrm{Re}\,\omega_1\big) \;+\; \big(1-\mathrm{Re}\,\omega_2\big)\,\Big].$$

For stability, η must lie in the unit ball of radius 1 centered at $(0,0)$ in the complex plane. That is,

$$0 \le 2\lambda\Big[\,\big(1-\mathrm{Re}\,\omega_1\big) \;+\; \big(1-\mathrm{Re}\,\omega_2\big)\,\Big] \le 2$$

Note that ω_1 and ω_2 are *arbitrary* complex numbers of magnitude 1. As a result the quantity in square brackets is a real number whose value may be anywhere in the interval $[0,4]$.

$$\max_{\omega_1,\omega_2} 2\lambda\Big[\,\big(1-\mathrm{Re}\,\omega_1\big) \;+\; \big(1-\mathrm{Re}\,\omega_2\big)\,\Big] = 8\lambda \le 2 \quad\Longrightarrow\quad \lambda \le \frac{1}{4}.$$

In higher dimensions, we keep adding more terms $1 - \mathrm{Re}\,\omega$, each of which has maximum value 2. Thus in d dimensions, the stability constraint for explicit Euler timestepping is

$$\lambda \;\le\; \frac{1}{2d} \qquad \text{in } d \text{ dimensions.}$$

As before, the implicit methods with $\frac{1}{2} \le \theta \le 1$ are stable for all λ.

The maximum growth rate is achieved when $\omega_1 = \omega_2 = -1$, so the spatial pattern is $(-1)^i(-1)^j$, a checkerboard.

Despite the $1/d$ effects described above, as you go to more dimensions, explicit methods become relatively more favored, for two reasons:

- The linear systems corresponding to the implicit methods become harder to solve. Explicit methods are just about as easy in three dimensions as in one, but implicit methods are much harder.

- It is not possible to take h small in high dimensions, because of the "curse of dimensionality:" the sheer numbers of grid points involved. If you can afford to put a million grid points in a domain whose edge length is size 10, then in 1-D you can have $h = 10^{-5}$, in 2-D you can have a 1000×1000 grid with $h = 10^{-2}$, and in 3-D you can have only $100 \times 100 \times 100$ for $h = 10^{-1} = 0.1$. Since the stability restriction depends only on the value of h, it becomes much less stringent.

The situation changes if you implement some sort of adaptive mesh method, which is able to achieve small grid steps h without requiring astronomical numbers of points. Unfortunately, in that case solving the linear system becomes even harder because it is ill-conditioned, but those problems are beyond the scope of a beginning course.

5 Alternating Direction Implicit methods

Here is one approach to avoiding solving the linear system for an implicit discretization on a rectangular grid. We need to introduce some notation.

The first-order differences of a grid function $u_{i,j}$ are

$$\begin{aligned} (\delta_x^+ u)_{ij} &= u_{i+1,j} - u_{i,j} & (\delta_y^+ u)_{ij} &= u_{i,j+1} - u_{i,j} \\ (\delta_x^- u)_{ij} &= u_{i,j} - u_{i-1,j} & (\delta_y^- u)_{ij} &= u_{i-1,j} - u_{i,j}. \end{aligned}$$

The second-order differences are

$$\begin{aligned} (\delta_x^2 u)_{i,j} &= (\delta_x^+ \delta_x^- u)_{i,j} = (\delta_x^- \delta_x^+ u)_{i,j} = u_{i+1,j} - 2u_{i,j} + u_{i-1,j} \\ (\delta_y^2 u)_{i,j} &= (\delta_y^+ \delta_y^- u)_{i,j} = (\delta_y^- \delta_y^+ u)_{i,j} = u_{i,j+1} - 2u_{i,j} + u_{i,j-1}. \end{aligned}$$

These are operators, so the second set literally means to apply first one then the other. They are essentially what is implemented in Matlab's `diff` operation, except for boundary effects which we are ignoring here. As noted above, these operators *commute* with each other, including mixtures of $x-$ and $y-$differences: $\delta_x^2 \delta_y^2$ is the same operation as $\delta_y^2 \delta_x^2$, etc.

Then our grid approximation to the Laplacian becomes

$$Lu = \frac{1}{h^2}\left(\delta_x^2 + \delta_y^2\right)$$

and our fully explicit approximation may be written

$$u^{n+1} \;=\; \Big(I \,+\, \lambda\delta_x^2 \,+\, \lambda\delta_y^2 \Big)\, u^n$$

The Crank-Nicolson scheme ($\theta = \frac{1}{2}$) may be written

$$\Big(I \,-\, \tfrac{1}{2}\lambda\delta_x^2 \,-\, \tfrac{1}{2}\lambda\delta_y^2 \Big)\, u^{n+1} \;=\; \Big(I \,+\, \tfrac{1}{2}\lambda\delta_x^2 \,+\, \tfrac{1}{2}\lambda\delta_y^2 \Big)\, u^n$$

ADI methods split the operator on the left and right sides, in a way that does not change the nature of the approximation very much, but makes the system much easier to solve. The simplest such is the *Peaceman-Rachford* scheme (1955):

$$\Big(I \,-\, \tfrac{1}{2}\lambda\delta_x^2 \Big) \Big(I - \tfrac{1}{2}\lambda\delta_y^2 \Big)\, u^{n+1} \;=\; \Big(I \,+\, \tfrac{1}{2}\lambda\delta_x^2 \Big) \Big(I + \tfrac{1}{2}\lambda\delta_y^2 \Big)\, u^n.$$

This is not quite the same as the Crank-Nicolson scheme, since an extra term $\frac{1}{4}\lambda^2\delta_x^2\delta_y^2 u^{n+1}$ has been added on the left, and $\frac{1}{4}\lambda^2\delta_x^2\delta_y^2 u^n$ has been added on the right. Thus the additional contribution to the truncation error is

$$\tfrac{1}{4}\lambda^2\delta_x^2\delta_y^2\delta_t^+ u \;\sim\; \tfrac{1}{4}\lambda^2 h^4 k\, u_{txxyy} \;\sim\; \mathcal{O}(k^3)$$

in which we have used $\delta_x^2 \sim h^2\partial_{xx}$, $\delta_y^2 \sim h^2 u\partial_{yy}$, and $\delta_t^+ u = u^{n+1} - u^n \sim k u_t$. Thus the additional error introduced by this splitting is of the same size as the truncation error $\mathcal{O}(k^3)$ already present in the second-order CN scheme: the modified scheme is still second-order accurate.

The Peaceman-Rachford modification is easily solved in two stages by using an intermediate variable $u^{n+1/2}$ and writing it as

$$\begin{aligned} \Big(I \,-\, \tfrac{1}{2}\lambda\delta_x^2 \Big)\, u^{n+\frac{1}{2}} \;&=\; \Big(I \,+\, \tfrac{1}{2}\lambda\delta_y^2 \Big)\, u^n \\ \Big(I \,-\, \tfrac{1}{2}\lambda\delta_y^2 \Big)\, u^{n+1} \;&=\; \Big(I \,+\, \tfrac{1}{2}\lambda\delta_x^2 \Big)\, u^{n+\frac{1}{2}}. \end{aligned}$$

To see that this is equivalent to the above unsplit form, operate on the first equation with $I + \frac{1}{2}\lambda\delta_x^2$ and on the second with $I - \frac{1}{2}\lambda\delta_x^2$, and use commutativity. Each of these two equations is a tridiagonal system and can be solved easily, accurately, and rapidly.

This modification does not destroy the stability properties of the Crank-Nicolson scheme, as may easily be checked by looking for exact solutions $u_{i,j}^n = \omega_1^i \omega_2^j \eta^n$. We find

$$\eta \;=\; \frac{\big(1 - \lambda(1 - \operatorname{Re}\omega_1)\big)\big(1 - \lambda(1 - \operatorname{Re}\omega_2)\big)}{\big(1 + \lambda(1 - \operatorname{Re}\omega_1)\big)\big(1 + \lambda(1 - \operatorname{Re}\omega_2)\big)}$$

instead of the CN form

$$\eta_{CN} \;=\; \frac{\big(1-\lambda(1-\operatorname{Re}\omega_1)-\lambda(1-\operatorname{Re}\omega_2)\big)}{\big(1+\lambda(1-\operatorname{Re}\omega_1)+\lambda(1-\operatorname{Re}\omega_2)\big)}$$

but both have $|\eta| \le 1$ for all λ.

6 Iterative solution methods

If you can't avoid the linear system, then you have to solve it. The "classical" methods apply to the matrix equality problem $Au = b$. For this standard problem, there are many methods available, both direct and iterative: for example, LU decomposition if A is tridiagonal.

We are now going to talk about "classical iterative methods." In practice, these methods have largely been superseded by more modern ones, such as conjugate gradient for symmetric problems, GMRES for asymmetric problems, or multigrid for problems arising from finite difference simulation.

The main idea is to choose some other matrix B, and write $Au = b$ as

$$Bu \;+\; (A-B)\,u \;=\; b.$$

We construct an sequence of approximate solutions $u^{(0)}, u^{(1)}, \ldots$ by

$$Bu^{(k+1)} \;+\; (A-B)\,u^{(k)} \;=\; b,$$

or

$$u^{(k+1)} \;=\; M\,u^{(k)} \;+\; B^{-1}b, \quad \text{with} \quad M = I - B^{-1}A.$$

If this sequence converges to a limit u_* then we know

$$u^{(k)} \to u_* \qquad \Longrightarrow \qquad Mu^{(k)} \to Mu_* \text{ and } u^{(k+1)} \to u_*.$$

And therefore

$$u_* \leftarrow u^{(k+1)} = Mu^{(k)} + B^{-1}b \to Mu_* + B^{-1}b.$$

This shows that u_* satisfies $u_* - Mu_* = B^{-1}b$ which is equivalent to satisfying $Au_* = b$.

Iterative methods never give the exact solution of $Au = b$ because the iteration cannot continue to infinity. What iterative methods can give you is something which is ϵ-close to the true solution. Of course, if you're using

matlab (or fortran or C) then you always have round-off error anyway and so you can't get the exact solution even if you used a direct method like Gaussian elimination or an LU decomposition.

In constructing an iterative method, we hope that the sequence will be easy to compute, and that it will converge rapidly. For this to be true, B must have two properties:

- B must be easily invertible; and
- B must be close to A, in the sense that M has small eigenvalues.

The problem of approximating a given matrix A by a "preconditioner" B which is easily invertible comes up in many situations. Of course, you still have to be able to apply A in the forward direction. (This may sound silly but for large, full matrices just doing matrix multiplication to compute $Au^{(k)}$ can be very slow.)

If $||M|| < 1$ then the iteration will certainly converge. If $e^{(k)} = u^{(k)} - u_*$ denotes the error, then $e^{(k+1)} = Me^{(k)}$, and so $\left|\left|e^{(k+1)}\right|\right| \leq ||M|| \left|\left|e^{(k)}\right|\right|$. It follows that $\left|\left|e^{(k)}\right|\right| \leq ||M||^k \left|\left|e^{(0)}\right|\right|$. This shows that $\left|\left|e^{(k)}\right|\right|$ goes to zero as $k \to \infty$, proving convergence.

The iteration will converge if and only if the *spectral radius of* M, $\rho(M)$ is less than 1

$$\rho(M) = max_{1 \leq i \leq n} |\lambda_i|$$

where M is an $n \times n$ matrix with real or complex entries. Having a spectral radius less than 1 implies convergence because as $k \to \infty$, $e^{(k)} \sim C\lambda_{max}^k v_{max}$, where λ_{max} is the dominant eigenvalue (the eigenvalue of largest magnitude) and v_{max} is the corresponding eigenvector.

The actual rate of convergence is thus given by $\rho(M)$ rather than $||M||$. Since $\rho(M) \leq ||M||$, this is a sharper and more informative condition — there are matrices for which $\rho(M) < 1 < ||M||$. For example,

$$\begin{pmatrix} \frac{1}{2} & 2 \\ 0 & \frac{1}{2} \end{pmatrix}$$

has $\rho(M) = 1/2$ and $||M|| = 9/4 + \sqrt{5}$. Here I took the L^2 norm of M:

$$||M|| = \sup_{\vec{x} \in \mathbb{R}^2} \frac{||M\vec{x}||}{||\vec{x}||} = \sup_{||\vec{x}||=1} ||M\vec{x}||$$

For matrices which arise from finite-difference approximation, you can sometimes get an acceptable approximate inverse B by splitting A into diagonal, lower-triangular, and upper-triangular parts:

$$A = D - L - U,$$

with D diagonal, L strictly lower-triangular, and U strictly upper-triangular (L and U have zeros on the diagonal).

Jacobi method Take $B = D$, so $M_J = D^{-1}(L+U)$. Of course D is trivally invertible since it is diagonal. The iteration may be written

$$u_i^{(k+1)} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1}^{i-1} A_{ij} u_j^{(k)} - \sum_{j=i+1}^{m} A_{ij} u_j^{(k)} \right).$$

It converges if A is *strictly* diagonally dominant: each diagonal element $|A_{ii}|$ is larger than either the sum of all the other elements in the same row or the sum in the same column: $\sum_{j \neq i} |A_{ij}|$ or $|A_{ii}| > \sum_{j \neq i} |A_{ji}|$. The Jacobi iteration does not change if you permute rows and columns of A and u, as long as you keep the same elements on the diagonal. (You can't gain anything by reordering the equations or the unknowns.)

Gauss-Seidel method Take $B = D - L$, so $M_{GS} = (D-L)^{-1}U$. Note that $D - L$ is lower-triangular, hence easily invertible by forward substitution. This is the same as the Jacobi formula, except that you replace elements of $u^{(k)}$ with elements of $u^{(k+1)}$ as soon as you have them, so you can use the same storage for both $u^{(k)}$ and $u^{(k+1)}$. It is written

$$u_i^{(k+1)} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1}^{i-1} A_{ij} u_j^{(k+1)} - \sum_{j=i+1}^{m} A_{ij} u_j^{(k)} \right). \tag{2}$$

where you compute $u_1^{(k+1)}, \ldots, u_m^{(k+1)}$ in order. The formula changes if you permute equations and variables, and it can sometimes be advantageous to explore rearrangements of the problem (*e.g,* "red-black" ordering in 2 or 3 dimensions). The Gauss-Seidel method converges under the same conditions (diagonal dominance) as the Jacobi method, but when they both converge, Gauss-Seidel always converges *faster*, typically twice as fast.