6 Higher order PDEs.

Classification of 2nd-order PDEs .

Consider the semi-linear 2nd-order PDE,

$$a(x,y)u_{xx} + 2b(x,y)u_{xy} + c(x,y)u_{yy} = F(x,y,u,u_x,u_y), \tag{6.1}$$

We introduce a change of variables,

$$\xi = \xi(x,y), \eta = \eta(x,y), \tag{6.2}$$

and attempt to simplify the equation. Three possibilities arise. If $b^2 - ac > 0$ then the equation can be reduced to the form (so-called normal form),

$$u_{\xi\eta} = D(\xi,\eta,u,u_\xi,u_\eta). \tag{6.3}$$

The equation is then called hyperbolic and it can be shown that ξ and η are characteristic variables. An alternative normal form is

$$u_{\xi\xi} - u_{\eta\eta} = D(\xi,\eta,u,u_\xi,u_\eta). \tag{6.4}$$

The wave equation

$$u_{xx} - c^2 u_{yy} = 0 \tag{6.5}$$

is hyperbolic.

If $b^2 - ac < 0$ then there are no real characteristics (technically they exist but are complex-valued). The normal form is

$$u_{\xi\xi} + u_{\eta\eta} = D(\xi,\eta,u,u_\xi,u_\eta). \tag{6.6}$$

Laplace's equation

$$u_{xx} + u_{yy} = 0 \tag{6.7}$$

is elliptic.

If $b^2 - ac = 0$ then the normal form is

$$u_{\xi\xi} = D(\xi,\eta,u,u_\xi,u_\eta). \tag{6.8}$$

The equation is then called parabolic.

The heat conduction equation

$$u_t = u_{xx} \tag{6.9}$$

is parabolic.

Classification in many cases is local. For example, in transonic aerodynamics, the Tricomi equation,

$$u_{xx} = xu_{yy}, \tag{6.10}$$

is hyperbolic in $x > 0$ and elliptic in $x < 0$.

The type of a PDE determines how we set up an initial/boundary value problem. For example, for a hyperbolic equation care must be taken when attempting to specify boundary conditions on a characteristic.

The wave equation.

Consider the classical wave equation for a function $u = u(x, t)$ with initial conditions given at time $t = 0$,

$$u_{tt} - c^2 u_{xx} = 0, \tag{6.11}$$

$$u(x, 0) = f(x), \quad u_t(x, 0) = g(x). \tag{6.12}$$

There is an exact general solution,

$$u = F(x - ct) + G(x + ct), \tag{6.13}$$

with arbitrary functions on the right which can be chosen to accommodate the initial conditions (6.12). The form of the exact solution confirms that the lines $x - ct = const$ and $x + ct = const$ are the two families of characteristics (information 'propagates' along characteristics). It makes sense physically. For acoustic waves, for instance, there are waves travelling along the $x-$ axis in both directions with the speed of sound, c.

Explicit method. Central derivatives.

Let $x_n = n\Delta x, t_m = m\Delta t$, and apply straightforward approximations for the second derivatives in the equation (6.11),

$$\begin{array}{llll}
\uparrow x & & & \\
 & \bullet(n+1, m) & & \\
\bullet(n, m-1) & \bullet(n, m) & \bullet(n, m+1) & \\
 & \bullet(n-1, m) & & \to t
\end{array}$$

$$\frac{u_{n,m+1} - 2u_{n,m} + u_{n,m-1}}{\Delta t^2} = c^2 \frac{u_{n+1,m} - 2u_{n,m} + u_{n-1,m}}{\Delta x^2}, \tag{6.14}$$

or, with $C = c\Delta t/\Delta x$,

$$u_{n,m+1} = -u_{n,m-1} + 2(1 - C^2)u_{n,m} + C^2(u_{n+1,m} + u_{n-1,m}). \tag{6.15}$$

Execrise. Check that the scheme is second order accurate with the LTE=$O((\Delta t)^2 + (\Delta x)^2)$.

Given the solution at two previous time levels, $m-1$ and m, the solution at time $m+1$ is computed straight away therefore the method is explicit.

Discussion. As in the leap-frog approximation for an ODE, there is a problem with the computation at the first time level. Try to introduce a virtual point with $m = -1$ and find the solution for $m = 1$ using a central approximation for the time derivative u_t at $t = 0$.

Stability.
Taking

$$u_{n,m} = \lambda^m e^{ikn}, \tag{6.16}$$

we get

$$\lambda = -\lambda^{-1} + 2 - 2C^2 + C^2(e^{ik} + e^{-ik}), \tag{6.17}$$

$$\lambda = -\lambda^{-1} + 2 - 2C^2 + 2C^2 \cos k, \tag{6.18}$$

$$\lambda = -\lambda^{-1} + 2[1 - C^2(1 - \cos k)], \tag{6.19}$$

$$\lambda = -\lambda^{-1} + 2[1 - 2C^2 \sin^2 \frac{k}{2}], \tag{6.20}$$

and finally a quadratic for λ,

$$\lambda^2 - 2A\lambda + 1 = 0, \tag{6.21}$$

where $A = 1 - 2C^2 \sin^2 \frac{kn}{2}$. The roots of the quadratic are

$$\lambda = A \pm \sqrt{A^2 - 1}. \tag{6.22}$$

If $|A| > 1$ then one of the roots has $|\lambda| > 1$ then the algorithm is unstable. When $|A| \leq 1$ then

$$|\lambda|^2 = 1, \tag{6.23}$$

and we have (neutral) stability. The condition of stability is therefore

$$-1 \leq 2C^2 \sin^2 \frac{k}{2} - 1 \leq 1, \tag{6.24}$$

which holds for any k if

$$C^2 \leq 1. \tag{6.25}$$

This is the standard CFL condition.

Exercise. Implicit scheme.

$$\begin{array}{lll}
\uparrow x & & \\
\bullet(n+1,m-1) & & \bullet(n+1,m+1) \\
\quad\bullet(n,m-1) & \bullet(n,m) & \quad\bullet(n,m+1) \\
\bullet(n-1,m-1) & & \bullet(n-1,m+1) \quad \rightarrow t
\end{array}$$

For the $x-$derivative we take the average value between the time steps $m+1$ and $m-1$ and obtain the following implicit algorithm,

$$\frac{u_{n,m+1} - 2u_{n,m} + u_{n,m-1}}{\Delta t^2} \tag{6.26}$$

$$= \frac{c^2}{2}\left(\frac{u_{n+1,m+1} - 2u_{n,m+1} + u_{n-1,m+1}}{\Delta x^2} + \frac{u_{n+1,m-1} - 2u_{n,m-1} + u_{n-1,m-1}}{\Delta x^2}\right). \tag{6.27}$$

Examine its stability.

Parabolic problems.

Diffusion equation with advection.
For a function $u = u(x,t)$, we have the following PDE,

$$u_t + a u_x = u_{xx}, \tag{6.28}$$

with constant a. The equation is parabolic, it generally requires an initial condition in time and two boundary conditions in x. The advection term au_x often signifies a background medium moving with speed a.

Explicit scheme, first order in t and second order in x.

$$\begin{array}{llll}
\quad\uparrow x & & & \\
\bullet(n+1,m) & & & \\
\quad\bullet(n,m) & \bullet(n,m+1) & & \\
\bullet(n-1,m) & & & \to t
\end{array}$$

$$\frac{u_{n,m+1}-u_{n,m}}{\Delta t}+a\frac{u_{n+1,m}-u_{n-1,m}}{2\Delta x}=\frac{u_{n+1,m}-2u_{n,m}+u_{n-1,m}}{\Delta x^2}, \tag{6.29}$$

or

$$u_{n,m+1}=\alpha u_{n+1,m}+\beta u_{n,m}+\gamma u_{n-1,m}, \tag{6.30}$$

where

$$\alpha=\frac{\Delta t}{\Delta x^2}-a\frac{\Delta t}{\Delta x},\quad \beta=1-2\frac{\Delta t}{\Delta x^2},\quad \gamma=\frac{\Delta t}{\Delta x^2}+a\frac{\Delta t}{\Delta x}. \tag{6.31}$$

To test for stability,

$$u_{n,m}=\lambda^m e^{ikn}, \tag{6.32}$$

then

$$\lambda=\alpha e^{ik}+\beta+\gamma e^{-ik}. \tag{6.33}$$

$$\lambda=1-2\left(1-\cos k\right)\frac{\Delta t}{\Delta x^2}-ia\frac{\Delta t}{\Delta x}\sin k. \tag{6.34}$$

As λ is complex valued we write $\lambda=\rho e^{i\Theta}$, for example, and then

$$\rho^2=\left(1-4\frac{\Delta t}{\Delta x^2}\sin^2\frac{k}{2}\right)^2+a^2\left(\frac{\Delta t}{\Delta x}\right)^2\sin^2 k. \tag{6.35}$$

In the case $a=0$ we have stability if

$$4\frac{\Delta t}{\Delta x^2}\sin^2\frac{k}{2}<2, \tag{6.36}$$

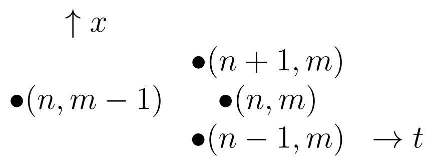
giving a sufficient condition on the size of the time step for a chosen spatial grid,

$$\Delta t<\frac{(\Delta x)^2}{2}. \tag{6.37}$$

Clearly, the explicit method is very restrictive on the step sizes in t if a high spatial resolution of the solution is needed.

Analysis for non-zero advection, $a\neq 0$, is somewhat harder and left as an exercise.

Implicit scheme, 1st order in time and 2nd order in space.
We consider equation (6.28) with $a = 0$ for simplicity.



Approximating the equation at the point (x_n, t_m) using a 1st-order backward time derivative we have,

$$\frac{u_{n,m} - u_{n,m-1}}{\Delta t} = \frac{u_{n+1,m} - 2u_{n,m} + u_{n-1,m}}{\Delta x^2}, \tag{6.38}$$

or

$$\alpha u_{n+1,m} + \beta u_{n,m} + \gamma u_{n-1,m} = -u_{n,m-1}, \tag{6.39}$$

$$\alpha = \gamma = \frac{\Delta t}{\Delta x^2}, \beta = -1 - 2\frac{\Delta t}{\Delta x^2}. \tag{6.40}$$

Stability. Testing on the usual solution $u_{n,m} = \lambda^m e^{ikn}$, we have

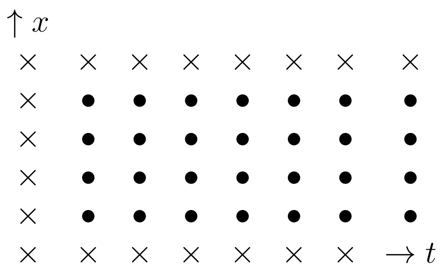
$$\lambda(\alpha e^{ik} + \beta + \gamma e^{-ik}) = -1, \tag{6.41}$$

and, after some manipulations,

$$\lambda^{-1} = 1 + 4\frac{\Delta t}{\Delta x^2}\sin^2\frac{k}{2}. \tag{6.42}$$

Therefore the implicit method is unconditionally stable.

Discussion. Organization of computations for the implicit method.



Despite very nice stability properties, the 1st-order in t algorithm is often too slow. An alternative 2nd-order method based on the Crank-Nicolson approximation is considered in a question sheet.

The Thomas algorithm.

Many problems lead to a boundary-value problem for a system of finite-difference equations of the form,

$$a_n u_{n+1} + b_n u_n + c_n u_{n-1} = d_n, \tag{6.43}$$

for the unknown $u_n, n = 0, ..., max$, with given coefficients a_n, b_n, c_n, d_n which could be numbers or square matrices. The end values, u_0 and u_{max} are given. In matrix form, this formulation reduces to an equation with a tri-diagonal matrix of coefficients. An efficient way of computing the solution relies on a form of Gauss elimination which can be implemented as follows. Let

$$u_{n+1} = p_n u_n + q_n, \tag{6.44}$$

with undetermined for now p_n, q_n. Substitution into (6.43) gives

$$u_n(a_n p_n + b_n) + c_n u_{n-1} = d_n, \tag{6.45}$$

or

$$u_n = -\frac{c_n}{a_n p_n + b_n} u_{n-1} + \frac{d_n}{a_n p_n + b_n}. \tag{6.46}$$

Comparing (6.46) with (6.44), we have

$$p_{n-1} = -\frac{c_n}{a_n p_n + b_n}, q_{n-1} = \frac{d_n - a_n q_n}{a_n p_n + b_n}. \tag{6.47}$$

At the last point,

$$u_{max} = p_{max-1} u_{max-1} + q_{max-1}, \tag{6.48}$$

and, since the value of u_{max} is known, this relation will hold for any value of u_{max-1}if

$$p_{max-1} = 0, \qquad q_{max-1} = u_{max}. \tag{6.49}$$

Now, using the recurrence formulae (6.47), the coefficients p_n and $q_n$are determined in a backward sweep for $n = max - 1, ..., 0$.

Finally, starting with the given value of u_0, the solution is recovered in the forward sweep, using (6.44) with the already computed p_n, q_n.

The stability and computational efficiency of the algorithm require special considerations.

Exercise. Develop an alternative procedure for a problem with the boundary conditions of the form

$$u_0 - \text{given}, \tag{6.50}$$

$$\alpha u_{max} + \beta u_{max-1} - \text{given}. \tag{6.51}$$

Such formulations arise, for example, in finite-difference approximations of a two-point boundary-value problem for an ODE with specified values of the function and its derivative at the ends of the solution interval.

Elliptic problems.
Laplace's equation.
Consider Laplace's equation in the $(x, y)-$plane,

$$u_{xx} + u_{yy} = 0. \tag{6.52}$$

In applications (e.g. potential flow theory, electrostatics) the solution is often sought in a domain D in the plane with boundary conditions specified on the domain boundary, ∂D, in one of three forms:
Dirichlet: the value of the function is specified on the boundary,

$$u(x, y) = f(x, y) \text{ on } \partial D; \tag{6.53}$$

Neumann: the normal to the boundary derivative of the function is specified on the boundary (given flux condition),

$$\frac{\partial u}{\partial n} = f(x, y) \text{ on } \partial D; \tag{6.54}$$

Robin: a linear combination of the function and its normal derivative is given on the boundary,

$$a(x, y)u(x, y) + b(x, y)\frac{\partial u(x, y)}{\partial n} = f(x, y) \text{ on } \partial D. \tag{6.55}$$

Exercise (quite hard). Laplace's equation is to be solved in $x \geq 0, -\infty < y < \infty$ subject to the initial conditions,

$$u|_{x=0} = f_0(y), \tag{6.56}$$

$$\left.\frac{\partial u}{\partial x}\right|_{x=0} = f_1(y). \tag{6.57}$$

Is this boundary-value formulation well posed?

Formulations for the Laplace equation in a half-plane are quite frequent in fluid dynamics. Note that in this example we are attempting to specify both the function and its derivative on the boundary at $x = 0$. To understand what is going on, either take the Fourier transform in y or consider simplified boundary conditions, for example $f_0 = e^{iky}, f_1 = 0$, and examine the behaviour of the solution with large positive k.

How should we alter the boundary-value formulation to make it well-posed?

Finite-difference schemes.

Five-point central differences.

Let $x_n = x_0 + n\Delta x, y_m = y_0 + m\Delta y$, then with the straightforward central approximations for the second derivatives we have,

$$\frac{u_{n+1,m} - 2u_{n,m} + u_{n-1,m}}{\Delta x^2} - \frac{u_{n,m+1} - 2u_{n,m} + u_{n,m-1}}{\Delta y^2} = 0. \tag{6.58}$$

The method is easy to understand and implement on a rectangular domain.

$$\begin{array}{llll}
\uparrow y & & & \\
 & & \bullet(n,m+1) & \\
 & \bullet(n-1,m) & \bullet(n,m) & \bullet(n+1,m) \\
 & & \bullet(n,m-1) & \\
 & & & \qquad \rightarrow x
\end{array}$$

Suppose also that the step sizes are equal, $\Delta x = \Delta y = h$. Then

$$u_{n-1,m} + u_{n,m-1} - 4u_{n,m} + u_{n+1,m} + u_{n,m+1} = 0. \tag{6.59}$$

There are several ways to solve the system of equations (6.59). Let us take Dirichlet boundary conditions with the value of u given on the boundary of a rectangular domain. By writing the unknown $u_{n,m}$ in a vector $\mathbf{u}$ the system (6.59) can be written in a matrix form,

$$A\mathbf{u} = \mathbf{b}, \tag{6.60}$$

where the vector $\mathbf{b}$ contains information from the nodes on the boundary. The matrix A is sparse as only five neighbouring points on the grid are linked via the finite-difference equation. Inverting A the answer is computed as $\mathbf{u} = A^{-1}\mathbf{b}$.

Exercise. Write down the matrix A and the vector $\mathbf{b}$ for the Dirichlet boundary-value problem. How is the form of A affected by the arrangement of $u_{n,m}$ into a vector? You may want to start with a small grid, 7 by 7, for example, before proceeding to the general case.

An alternative, and often preferred method is to solve (6.59) using iterations.

Relaxation methods.

Relaxation or iterative methods lead to a sequence of approximations to the exact solution of (6.60). The idea is to write the matrix A as

$$A = E - F, \tag{6.61}$$

where E is easily invertible. Then rearranging the equation,

$$E\mathbf{u} = F\mathbf{u} + \mathbf{b}, \tag{6.62}$$

iterations are computed starting from an initial guess, $\mathbf{u}^0$, according to

$$\mathbf{u}^{l+1} = E^{-1}(F\mathbf{u}^l + \mathbf{b}), \tag{6.63}$$

where $l = 0, 1, 2, ...$ is the number of iteration. Examples are point Jacobi and Gauss-Seidel methods below.

Point Jacobi iterations.

$$\begin{array}{ccccccc}
(1, m_{max})\times & \times & \times & \times & \times & \times & (n_{max}, m_{max})\times \\
\times & \bullet & \bullet & \bullet & \bullet & \bullet & \times \\
\times & \bullet & \bullet & \bullet & \bullet & \bullet & \times \\
\times & \bullet & \bullet & \bullet & \bullet & \bullet & \times \\
\times & \bullet & \bullet & \bullet & \bullet & \bullet & \times \\
\times & \bullet & \bullet & \bullet & \bullet & \bullet & \times \\
(1, 1)\times & \times & \times & \times & \times & \times & (n_{max}, 1)\times
\end{array}$$

The value of u is given at the nodes labelled with a cross (the numbering for the corner points is also shown). The task is to find $u_{n,m}$ for the nodes marked as dots. Suppose an initial guess is made on the distribution of $u_{n,m}$ for all n and m. Call this initial guess $u^0_{n,m}$. Using (6.59), we compute updated values of $u_{n,m}$ for all nodes (without changing any values at the boundaries) using the following formula,

$$u^0_{n-1,m} + u^0_{n,m-1} - 4u^1_{n,m} + u^0_{n+1,m} + u^1_{n,m+1} = 0, \tag{6.64}$$

or, if you prefer,

$$u^1_{n,m} = \frac{u^0_{n-1,m} + u^0_{n,m-1} + u^0_{n+1,m} + u^0_{n,m+1}}{4}, \tag{6.65}$$

for $2 \leq n \leq n_{max} - 1, 2 \leq m \leq m_{max} - 1$. The last form shows that the update algorithm is simply an averaging procedure over the neighbouring nodes. (This is closely related to the mean value theorem for harmonic functions.)

Once $u^1_{n,m}$ is computed for all internal nodes, the process is repeated, using the updated values as the new initial distribution. We then have the following iterative procedure, with l denoting the number of the iteration,

$$u^{l+1}_{n,m} = \frac{u^l_{n-1,m} + u^l_{n,m-1} + u^l_{n+1,m} + u^l_{n,m+1}}{4}. \tag{6.66}$$

Iterations terminate when the difference between successive iterations is sufficiently small (hoping that this difference can be made zero as the number of iterations tends to infinity in which case iterations are called convergent).

Convergence.

A quick test on convergence is done by assuming

$$u^l_{n,m} = \rho^l e^{i(k_1 n + k_2 m)}, \tag{6.67}$$

with real k_1, k_2 and $\rho = \rho(k_1, k_2)$ to be found. Substitution into (6.66) yields,

$$\rho = \frac{1}{4}(e^{-ik_1} + e^{-ik_2} + e^{ik_1} + e^{ik_2}), \tag{6.68}$$

$$\rho = \frac{1}{2}(\cos k_1 + \cos k_2), \tag{6.69}$$

which shows that $|\rho| \leq 1$ therefore the iterations converge albeit slowly (especially when both $\cos k_{1,2} \approx 1$).

Gauss-Seidel iterations.

We can speed up convergence if, at each iteration, we engage already updated nodes in the update of $u_{n,m}$. This can be done in several ways depending on the chosen sequence of updating the nodes (refer to the grid to visualize the process by updating in horizontal rows starting from the bottom left corner, or in vertical columns starting from the top left). Such algorithms are called Gauss-Seidel. As an example, consider the following,

$$u^l_{n+1,m} + u^{l+1}_{n-1,m} - 4u^{l+1}_{n,m} + u^{l+1}_{n,m+1} + u^l_{n,m-1} = 0, \tag{6.70}$$

which implies that the nodes above and left of the pivot node (n, m) need to be updated before we can update $u_{n,m}$. Gauss-Seidel iterations tend to converge faster.

Successive over-relaxation (SOR).

Although SOR techniques are most often used in conjunction with the Gauss-Seidel iterations, we shall introduce the idea in a more simple, Jacobi, iterative procedure. Jacobi iterations (6.66) can be written as

$$u^{l+1}_{n,m} = F^l, \tag{6.71}$$

with

$$F^l = \frac{1}{4}(u^l_{n+1,m} + u^l_{n-1,m} + u^l_{n,m+1} + u^l_{n,m-1}). \tag{6.72}$$

Let us write this as

$$u^{l+1}_{n,m} = u^l_{n,m} + \Delta^l, \tag{6.73}$$

where the residual, Δ^l, is given by $\Delta^l = F^l - u^l_{n,m}$ so that at each iteration the value of $u_{n,m}$ is corrected by an amount equal to the residual.

Suppose we want to accelerate the convergence by adding a bigger fraction of the residual at each iteration, that is we take,

$$u^{l+1}_{n,m} = u^l_{n,m} + \omega\Delta^l, \tag{6.74}$$

with some constant ω which is called the relaxation parameter. Then

$$u^{l+1}_{n,m} = (1 - \omega)u^l_{n,m} + \omega F^l. \tag{6.75}$$

If $\omega > 1$ we have over-relaxation, if $\omega < 1$ the procedure is called under-relaxation. In some cases, under-relaxation helps when the straightforward iterative procedure is divergent.

Discussion: optimal relaxation parameter.

Exercise. Using substitution (6.67), examine the convergence of the SOR algorithm (6.75), (6.72) for small and positive values of k_1, k_2.