MAST90026 Computational Differential Equations: Week 8

Jesse Collis
Modified from Hailong Guo (2022)

Semester 1 2024

The University of Melbourne

Evolution PDEs

Classical examples:

$$
\begin{aligned}
&u_t = u_{xx}, && \text{diffusion equation,}\\
&u_t + cu_x = 0, && \text{transport equation,}\\
&u_{tt} = c^2 u_{xx}, && \text{wave equation,}
\end{aligned}
$$

Focus on 1+1 D (i.e. time + x) problems and 1+2D (i.e. time $+x, y$. 1+3 D problems can be solved using the same techniques just trickier implementation.

We can think of 1+1 D PDEs as adding time to a BVP or adding space to an IVP. To be well posed, typically we will need both boundary conditions (like a BVP) and initial conditions (like an IVP).

Parabolic equation

Consider

$$u_t - (D(x)u')' + q(x)u = r(x),$$

with

$$\begin{aligned} \text{BCs} : u(a,t) &= \alpha(t), \\ u(b,t) &= \beta(t), \\ \text{IC} : u(x,0) &= u_0(x). \end{aligned}$$

If $D = 1$, $q = 0$, $r = 0$ then this is the heat/diffusion equation.

Two classes of numerical methods:

- Semi-discretisation
- Full discrete

Semi-discretisation (Method of line)

Idea: discretise in space only, e.g. FD, FEM, finite volume, spectral, collocation .etc. as before. This produces a system of ODEs that we solve with an IVP package e.g. *ode*45 in **Matlab**.

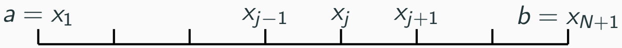
For larger problems, we need to discretise in time as well, e.g. if our IVP solver cannot handle the system of ODEs. This gives *fully discrete methods*.

But for modest problems the Method of Lines (MoL) is a good approach.

Semi-discretisation: Finite difference method for space

Consider the problem

$$u_t - (D(x)u_x)_x = r(x,t).$$



When we have an equation in flux form (or conservative form) e.g. from a conservation law, it's good practice to discretize each derivative separately, rather than expand derivatives and then discretize, i.e. $(D(x)u_x)_x$ rather than $D'(x)u_x + D(x)u_{xx}$.

Semi-discretisation: Finite difference scheme

Discretise $-(D(x)u_x)_x$ using central differences. We can use FD on $-D(x)u_x$ directly without expanding it.

$$
\begin{aligned}
D(x)u_x\big|_{x_j} &\approx \frac{D(x_j)(u_{j+1/2} - u_{j-1/2})}{h}, \\
(D(x)u_x)_x\big|_{x_j} &\approx \frac{D(x)u_x\big|_{x_{j+1/2}} - D(x)u_x\big|_{x_{j-1/2}}}{h} \\
&= \frac{1}{h}\left(\frac{D(x_{j+1/2})(u_{j+1} - u_j)}{h} - \frac{D(x_{j-1/2})(u_j - u_{j-1})}{h}\right) \\
&= \frac{1}{h^2}\left(D(x_{j+1/2})u_{j+1} - (D(x_{j+1/2}) + D(x_{j-1/2}))u_j + D(x_{j-1/2})u_{j-
\end{aligned}
$$

Semi-discretisation: Dirichlet boundary condition

Add Dirichlet boundary conditions:

$$u_1(t) = \alpha(t), \qquad u_{N+1}(t) = \beta(t).$$

Therefore at positions $j = 2, \dots, N$ we have the following equations:

$$\dot{u}_j(t) - \frac{1}{h^2}\left(D(x_{j+1/2})u_{j+1} - (D(x_{j+1/2}) + D(x_{j-1/2}))u_j + D(x_{j-1/2})u_{j-1}\right) = r(x_j,$$

This is a complete system of $N-1$ ODEs. Use the BCs to modify the equations for $\dot{u}_2$ and $\dot{u}_N$. The initial condition becomes $u_j(0) = u_0(x_j)$.

Question: how big is the spatial discretisation error?
Answer: $O(h^2)$ because it's central differences

Semi-discretisation: Finite element method for space

Integrate by parts to get the weak formulation of the problem:

$$\int_a^b u_t\, v\, dx + \int_a^b D(x)\, u_x\, v_x\, dx = \int_a^b r\, v\, dx + \left(D(x)\, u_x\, v\right)\Big|_a^b, \forall v \in H_0^1(a,b).$$

Choose

$$\begin{aligned} u &= U_0(x) + \bar{u} \quad \in H_0^1 \times \mathbb{R}^+ \\ &= U_0(x) + \sum u_j(t) \underbrace{\phi_j(x)}_{\text{nodal basis}} \quad . \end{aligned}$$

Note: We need U_0 only for theoretical purpose. In the implementation, we don't need to construct U_0.

Semi-discretisation: Finite element method for space

Now we get the Galerkin equations. Replace v above with ϕ_k

$$\int_a^b \left(\sum_j \dot{u}_j \phi_j \right) \phi_k \, dx + \int_a^b D(x) \left[U_0' + \sum_j u_j(t) \phi_j' \right] \phi_k' \, dx = \int_a^b r \, \phi_k \, dx.$$

Swapping the order of the sum and the integral:

$$\sum_j \underbrace{\left(\int_a^b \phi_j \phi_k \, dx \right)}_{\mathbf{M}} \dot{u}_j + \sum_j \underbrace{\left(\int_a^b D(x) \phi_j' \phi_k' \, dx \right)}_{\mathbf{K}} u_j$$

$$= \underbrace{\int_a^b r \phi_k \, dx - \int_a^b D(x) U_0' \phi_k' \, dx}_{\mathbf{f}}.$$

In matrix and vector notation:

$$\mathbf{M}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f}.$$

Recap of ODE solvers

Four classes of modern methods used to solve this:

1. **Explicit Runge-Kutta methods (1895–1905)**: Simple to code, one-step methods. Matlab's $ode23$, $ode45$ are based on this.
2. **Explicit multi-step methods:** The most popular is Adams (1855). Very efficient, cheap higher-order methods (requiring one function evaluation per step, vs 3 or 6 for RK). Matlab: $ode113$.
3. **Implicit multi-step methods:** Most famous is Gear's Backward Differentiation Formula (1971). Matlab: $ode15s$.
4. **Implicit Runge-Kutta methods:** Have nice mathematical properties but are computationally expensive. Matlab: $ode23t$, $ode23tb$. There is also the TRBDF2 algorithm (1985) used for circuit simulation.

Note: Need special solvers (e.g. $ode15s$) for Stiff problems: slow and fast time scales present in the same problem

Recap of ODE solvers: Linear stability analysis

Consider the autonomous case (i.e., where f has no explicit dependence on t):

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}).$$

We travel along a solution $\bar{\mathbf{u}}$, varying slowly. In each step, we introduce some error $\mathbf{z}$. How does this error propagate?

Write $\mathbf{y}(t) = \bar{\mathbf{u}} + \mathbf{z}(t)$, so:

$$\begin{aligned} & \dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}), \\ \implies \quad & \dot{\mathbf{z}} = \mathbf{f}(\bar{\mathbf{u}} + \mathbf{z}) \\ & \quad \approx \mathbf{f}(\bar{\mathbf{u}}) + \left.\frac{\partial \mathbf{f}}{\partial \bar{\mathbf{u}}}\right|_{\bar{\mathbf{u}}} \mathbf{z} + O(||\mathbf{z}||^2). \end{aligned}$$

So locally this behaves like $\dot{\mathbf{z}} = J|_{\bar{\mathbf{u}}}\mathbf{z} + \mathbf{b}$

Recap of ODE solvers: Linear stability analysis

If J is diagonalisable (has a full set of linearly independent eigenvectors), we can write $J = T\Lambda T^{-1}$ (by the spectral theorem) where Λ is a diagonal matrix of eigenvalues and T is a matrix of eigenvectors. So:

$$\begin{gathered} \dot{\mathbf{z}} = T\Lambda T^{-1}\mathbf{z} + \mathbf{b}, \\ \text{Define } \mathbf{w} = T^{-1}\mathbf{z}, \text{ or } \mathbf{z} = T\mathbf{w}, \\ \text{So } \dot{\mathbf{w}} = \Lambda\mathbf{w} + T^{-1}\mathbf{b}. \end{gathered}$$

This is a system of uncoupled scalar ODEs, $\dot{w}_i = \lambda_i w_i + b_i$. Since a real matrix can have complex eigenvalues, $\lambda \in \mathbb{C}$.

Recap of ODE solvers: Linear stability analysis

Note we use superscript to denote time, i.e. $w^n \approx w(t_n)$ and k denote time step size, $k = \Delta t$.

Region of Absolute Stability (RAS):

$$\{\lambda k \in \mathbb{C} : |w^n| \text{ stays bounded as } n \to \infty\}.$$

i.e., for a given eigenvalue, what time step do we need so error does not blow up.

Recap of ODE solvers: Linear stability analysis

A-stability: $|w^n|$ bounded whenever $\mathbb{R}(\lambda) < 0$.

i.e., if true solution decays, error remains bounded.

L-stability: $\left|\frac{w^{n+1}}{w^n}\right| \to 0$ as $\lambda \to -\infty$.

i.e., for fastest possible decay, error decays to zero.

Stronger than A-stability. Desirable when solving stiff problems.

Recap of ODE solvers: Euler method

RAS:

$$\begin{aligned} w^{n+1} &= w^n + kf(t_n, w^n) \\ &= w^n + k\lambda w^n \\ &= w^n(1 + k\lambda), \\ \left|w^{n+1}\right| < \infty \text{ as } n \to \infty &\Rightarrow |1 + k\lambda| \le 1, \\ &\Rightarrow 0 \le k \le 2/|\lambda|. \end{aligned}$$

If $\lambda \approx -1$ stability is not an issue, but still need k smaller for accuracy reasons. But what if $\lambda \ll -1$? Then we need $k < \left|\frac{2}{\lambda_{\max}}\right| \ll 1$. Now stability requirements force a much smaller k than accuracy requirements.

Heat equation

Consider $u_t = u_{xx}$ using finite difference

$$\begin{bmatrix} \frac{du_2}{dt} \\ \frac{du_3}{dt} \\ \vdots \\ \frac{du_N}{dt} \end{bmatrix} = \underbrace{\frac{1}{h^2} \begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & & 1 & -2 \end{bmatrix}}_{A} \begin{bmatrix} u_2 \\ u_3 \\ \vdots \\ u_N \end{bmatrix} + \begin{bmatrix} \frac{\alpha(t)}{h^2} \\ 0 \\ \vdots \\ \frac{\beta(t)}{h^2} \end{bmatrix}$$

or

$$\dot{\mathbf{u}} = A\mathbf{u} + \mathbf{b}$$

where $\mathbf{u} = \begin{bmatrix} u_2 & u_3 & \dots & u_N \end{bmatrix}^T$ and $\mathbf{b} = \begin{bmatrix} \frac{\alpha(t)}{h^2} & 0 & \dots & \frac{\beta(t)}{h^2} \end{bmatrix}^T$

Heat equation: eigen analysis

A has eigenvalues $\frac{2}{h^2}(\cos(\pi jh) - 1)$, for $j = 1, \dots, N-1$.

$$\begin{aligned}
\lambda_1 &= \tfrac{2}{h^2}\left(\cos(\pi h) - 1\right) \\
&= \tfrac{2}{h^2}(1 - \tfrac{1}{2}\pi^2 h^2 + \cdots - 1) \\
&= -\pi^2 + O(h^2), \\
\lambda_{N-1} &= \tfrac{2}{h^2}\left(\cos(\pi(N-1)h) - 1\right) \\
&\approx \tfrac{2}{h^2}(-1 - 1) \\
&= -4/h^2.
\end{aligned}$$

λ_1 is bounded away from zero, so $\|A^{-1}\|_2 \leq 1/\pi^2$; whereas $\lambda_{N-1} \to -\infty$ as $h \to 0$. Thus:

$$\kappa_2(A) = \left|\frac{\lambda_{\max}}{\lambda_{\min}}\right| \approx \frac{4}{\pi^2 h^2}.$$

Fully discrete: Parabolic PDEs

Idea: discretise the temporal derivative we get a set of fully discrete methods.

Semi-discretisation of $u_t = u_{xx}$ using finite difference: $\dot{\mathbf{u}} = A\mathbf{u} + \mathbf{b}$.

FTCS(Forward time central space):

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \frac{k}{h^2}\, h^2 A\mathbf{u}^n + k\mathbf{b}^n.$$

Discretisation error is LTE $= \mathcal{O}(k, h^2)$, i.e. the method is first order in time and second order in space.

Fully discrete: Stability analysis

Eigenvalues of A lie in $[-4/h^2, -\pi^2] \Rightarrow |\lambda_i|_{\max} \approx 4/h^2$.

For stability we need $k\lambda_i$ to lie within the RAS for the time-stepping scheme. Euler's method has RAS $|1 + k\lambda| < 1$.

Substitute λ to get the stability of FTCS:

$$\begin{aligned} \left|1 + k\left(\tfrac{-4}{h^2}\right)\right| &< 1, \\ \left|1 - \tfrac{4k}{h^2}\right| &< 1, \\ -1 < 1 - \tfrac{4k}{h^2} &< 1, \\ \implies \quad -\tfrac{4k}{h^2} &< 0, \qquad \text{tells us nothing,} \\ \text{and} \qquad \tfrac{k}{h^2} &< 1/2, \qquad \text{i.e. } r < 1/2. \end{aligned}$$

$\Rightarrow$ FTCS is conditionally stable.

This is a severe restriction on the time step. If $h = 0.01$, $k < 5 \times 10^{-5}$. That is a very small time step!

Backward Euler Method

Backward Euler method:

$$\begin{aligned} \dot{w}\big|_{t_n} &= \frac{w^n - w^{n-1}}{k}, \\ \implies \quad w^n &= w^{n-1} + k\, f(t_n, w^n). \end{aligned}$$

RAS:

$$\begin{gathered} \dot{w} = \lambda w \quad \implies \quad w^{n+1} = w^n + k\lambda w^{n+1}, \\ w^{n+1}(1 - k\lambda) = w^n, \\ \left|\tfrac{w^{n+1}}{w^n}\right| \text{ bounded if } \left|\tfrac{1}{1-k\lambda}\right| < 1, \\ \text{i.e. } |1 - k\lambda| > 1. \end{gathered}$$

$\Rightarrow$ A-stable and L-stable (as $\lambda \to \infty$, $\left|\frac{w^{n+1}}{w^n}\right| = \left|\frac{1}{1-k\lambda}\right| \to 0$).

Fully discrete: BTCS

Semi-discretisation of $u_t = u_{xx}$ using finite difference: $\dot{\mathbf{u}} = A\mathbf{u} + \mathbf{b}$.

BTCS:

$$\left(I - \frac{k}{h^2} h^2 A\right) \mathbf{u}^{n+1} = \mathbf{u}^n + \bar{\mathbf{b}}^{n+1}.$$

The matrix $I - kA$ is tridiagonal so solving is cheap.

Summary of BTCS:

- all eigenvalues of A are in RAS
- BTCS unconditional stable
- Choose k, h for accuracy reason
- LTE $= \mathcal{O}(k, h^2) \Rightarrow k \sim h^2$ to balance accuracy and computational effort.

Fully discrete: Crank-Nicolson Method

Trapezoid rule for solving an ODE:

$$w^{n+1} = w^n + \tfrac{1}{2}k(f(t_n, w^n) + f(t_{n+1}, w^{n+1})).$$

This has $O(k^2)$ time step error.

Crank-Nicolson Method:

$$\left(I - \frac{k}{2h^2}h^2 A\right)\mathbf{u}^{n+1} = \left(I - \frac{k}{2h^2}h^2 A\right)\mathbf{u}^n + \frac{k}{2}(\mathbf{b}^n + \mathbf{b}^{n+1}).$$

Unconditional stable and LTE $= \mathcal{O}(k^2, h^2) \Rightarrow$ Default method for diffusion equation.

Von Neumann analysis

Another way to get stability restriction without knowing e-values of matrix A.

Von Neumann analysis is based Fourier analysis and hence is generally limited to constant coefficient PDEs. For simplicity, consider unbounded spatial domain.

Use a Fourier transform to solve a linear PDE on $\mathbb{R}$:

$$\frac{\partial}{\partial x} e^{iqx} = iqe^{iqx}.$$

$\Rightarrow w(x) = e^{iqx}$ is an eigenfunction of $\frac{\partial}{\partial x}$ operator.

For any difference operator (FD, BD, CD), we get a grid function $W_j = e^{iqx_j} = e^{iqjh}$ which is an eigenfunction of the difference operator.

Von Neumann analysis: eigenfunctions

Forward differences:

$$\begin{aligned}\frac{W_{j+1} - W_j}{h} = \frac{e^{iq(j+1)h} - e^{iqjh}}{h} &= \frac{1}{h} e^{iqjh}(e^{iqh} - 1) \\ &= e^{iqjh} \underbrace{\left(\frac{e^{iqh} - 1}{h} \right)}_{\text{eigenvalue}},\end{aligned}$$

Relationship between eigenfunction of forward difference and eigenfunction of $\frac{\partial}{\partial x}$.

Von Neumann analysis: discrete Fourier transform

Suppose we have a grid function V_j defined at grid points $x_j = jh$ for $j = 0, \pm 1, \pm 2, \ldots$, which is an l_2 function in the sense that the 2 -norm

$$\|U\|_2 = \left(h \sum_{j=-\infty}^{\infty} |U_j|^2 \right)^{1/2}$$

Express V_j as a linear combination of the grid funcitons e^{ijhq}.

$$V_j = \frac{1}{2\pi} \int_{-\pi/h}^{\pi/h} \hat{V}(q)\, e^{iqhj}\, dq,$$

where

$$\hat{V}(q) = h \sum_{j=-\infty}^{\infty} e^{-iqhj}\, V_j.$$

Von Neumann analysis: Parseval's relation

Parseval's Relation: Using the grid function 2-norm,

$$\|u\|_{\ell^2} = \left(h \sum_j |u_j|^2 \right)^{1/2} = \|\hat{U}\|_2 = \left(\int_{-\pi/h}^{\pi/h} |\hat{U}(q)|^2 \, dq \right)^{1/2}.$$

For strong stability, we want to show that

$$\|u^{n+1}\|_{\ell^2} \leq \|u^n\|_{\ell^2}, \qquad \text{i.e.} \quad \|\hat{U}^{n+1}\|_2 \leq \|\hat{U}^n\|_2.$$

Von Neumann analysis: FTCS

Look for $\hat{U}^{n+1}(q) = g(q)\hat{U}^n(q)$, where $g(q)$ is an "amplification factor".

To find $g(q)$ we use the eigenfunction e^{iqjh}:

$$\begin{aligned} u_j^{n+1} &= u_j^n + r(u_{j+1}^n - 2u_j^n + u_{j-1}^n), \\ u_j^{n+1} &= e^{iqhj} + r(e^{iqh(j+1)} - 2e^{iqhj} + e^{iqh(j-1)}) \\ &= e^{iqjh}(1 + re^{iqh} - 2 + e^{-iqh}) \\ &= u_j^n \underbrace{(1 + 2r(\cos(qh) - 1))}_{= \, g(qh)}. \end{aligned}$$

For strong stability, we want $|g(qh)| < 1$ for $qh \in [-\pi, \pi]$. So we need:

$$\begin{aligned} 1 - 4r < g = 1 + 2r(\cos(qh) - 1) < 1, \\ 1 - 4r > -1, \\ r < 1/2. \end{aligned}$$

Use Von Neumann stability analysis to show BTCS and Crank-Nisolson method are unconditionally stable.

$1+2$ D parabolic problem

Consider 1+2D parabolic problem:

$$u_t = u_{xx} + u_{yy}.$$

Using Crank-Nicolson plus a five-point stencil, we end up with equations like:

$$\begin{aligned} u_{ij}^{n+1} &= u_{ij}^n + \tfrac{k}{2}\left(\nabla_h^2 u_{ij}^n + \nabla_h^2 u_{ij}^{n+1}\right), \\ \implies \quad \underbrace{(I - \tfrac{k}{2}\nabla_h^2)}_{=\,A} u_{ij}^{n+1} &= (I + \tfrac{k}{2}\nabla_h^2) u_{ij}^n. \end{aligned}$$

condition number $\kappa_2(A) = O(k/h^2) \Rightarrow$ better conditioned than Poisson equation $\Rightarrow$ converge faster

$1+2$ D parabolic problem: locally one-dimensional(LOD)

Idea: Use the fact that $\nabla_h^2 = D_x^2 + D_y^2$ to break up the computation: solve in x direction, then in y.

$$
\begin{aligned}
u_{ij}^{n+1} &= u_{ij}^n + \tfrac{k}{2}(D_x^2 u_{ij}^n + D_x^2 u_{ij}^{n+1} + D_y^2 u_{ij}^n + D_y^2 u_{ij}^{n+1}),\\
\implies \quad u_{ij}^* &= u_{ij}^n + \tfrac{k}{2}(D_x^2 u_{ij}^n + D_x^2 u_{ij}^*),\\
u_{ij}^{n+1} &= u_{ij}^* + \tfrac{k}{2}(D_y^2 u_{ij}^* + D_y^2 u_{ij}^{n+1}),\\
\implies \quad (I - \tfrac{k}{2}D_x^2)u^* &= (I + \tfrac{k}{2}D_x^2)u^n, \qquad \text{C–N in } x, \qquad (1)\\
(I - \tfrac{k}{2}D_y^2)u^{n+1} &= (I + \tfrac{k}{2}D_y^2)u^*, \qquad \text{C–N in } y. \qquad (2)
\end{aligned}
$$

In (1), u^* is compiled over rows through D_x^2. Each equation for j is independent, so there are $j = 0, \dots, m+1$ systems, and each system has a tridiagonal matrix for u_{ij}^*. So it's $O(m^2)$ operations to solve for u^*.

In (2), u^{n+1} is compiles over columns through D_y^2, giving m tridiagonal systems. Total work is $2m+2$ tridiagonal systems of size m, for $O(m^2)$ operations.

$1+2$ D parabolic problem: alternative direction implicit(ADI)

In the first step, the y diffusion is explicit, x diffusion is implicit. In the second step, it is reversed.

$$\begin{aligned} u_{ij}^* &= \tfrac{k}{2}(D_y^2 u_{ij}^n + D_x^2 u_{ij}^*), \\ u_{ij}^{n+1} &= u_{ij}^* + \tfrac{k}{2}(D_x^2 u_{ij}^* + D_y^2 u_{ij}^{n+1}). \end{aligned}$$

$\mathcal{O}(k^2, h^2)$ error, unconditionally stable, $O(m^2)$ operations per time step.