Lecture 10

MTH572/MTH472
Numerical Methods for PDEs
Alfa Heryudono

UMass Dartmouth

Main references (quotes):
Trefethen: NumPDE, ATAP, Spectral Methods in MATLAB
Fornberg: PS Guide
Leveque: NumPDE
Driscoll: Learning MATLAB

Solving time-dependent PDE: Method of Lines

A common and popular technique to numerically solve time-dependent PDEs is by using a method called

Method of Lines.

- Discretize the space with Finite-Difference (or some other methods)
- March the resulting system of ODEs in time using Linear Multistep or Runge-Kutta methods.

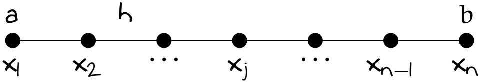
Let's take a 1-D advection equation

$$\begin{aligned} \text{PDE:} \quad & \frac{\partial u}{\partial t} = \frac{\partial u}{\partial x} \quad \text{for } x \in [a, b] \\ \text{IC:} \quad & u(x, 0) = u_0(x) \quad \text{when } t = 0 \\ \text{BC:} \quad & u(b) = g(t) \quad \text{at } x = b \end{aligned}$$

As a simple example, we want to

- Discretize the interval $[a, b]$ with n equally-spaced points.
- Use 3-point stencil based FD differentiation matrix.
- Advance the solution in time with Forward Euler.

Step 1: Discretize the interval $[a, b]$ with n equally-spaced points.



Step 2: Approximate $\frac{\partial u}{\partial x}$ with 3-point stencil FD. For simplicity, we can first form the system of ODEs all the way to the boundary.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix}_t = \frac{1}{2h} \begin{bmatrix} -3 & 4 & -1 & \cdots & 0 \\ -1 & 0 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 0 & 1 \\ 0 & \cdots & 1 & -4 & 3 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix}$$

Now we need to remove the n-th row since the PDE is only enforced at $x_1, \ldots, x_{n-1}$. The boundary condition at $x = x_n$ is $u_n = g(t)$.

Step 3: Strip the last row and split the last column that is affected by BC value separate from the system that contains the unknowns. Note that $u_n = g(t)$.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{bmatrix}_t = \frac{1}{2h} \begin{bmatrix} -3 & 4 & -1 & \cdots & 0 \\ -1 & 0 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 0 & 1 \\ 0 & \cdots & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{bmatrix} + \frac{g(t)}{2h} \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

Step 4: March the system of ODEs in time

$$\underline{u}_t = \mathbf{L}\underline{u} + \underline{f}(t) \quad \text{with initial condition at } t = 0 \text{ to be } \underline{u}_0$$

with Forward Euler Method, where

$$\underline{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{bmatrix} \quad \mathbf{L} = \frac{1}{2h} \begin{bmatrix} -3 & 4 & -1 & \cdots & 0 \\ -1 & 0 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 0 & 1 \\ 0 & \cdots & 0 & -1 & 0 \end{bmatrix} \quad \underline{f}(t) = \frac{g(t)}{2h} \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

Things to do in class

1. With time-step $k = 10^{-2}$, write the MATLAB code (with $n = 50$) to simulate

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x} \text{ for } x \in [-1, 1)$$
$$u(x, 0) = \sin(\pi x) \text{ (IC)}, \quad u(1) = 0 \text{ (BC)}$$

   from $t = 0$ until $t = 1$. Redo the experiment with $k = 10^{-1}$. What will happen to the solution ?
2. Compute the eigenvalues Λ of the matrix $\mathbf{L}$ and plot them on the same figure along with the stability region of the FE method. Is it possible to scale the time-step k such that $k\Lambda$ are inside the stability region of the FE ?
3. Redo problem 1, 2 with Backward Euler, AB2, AM2, and BDF2.
4. With $n = 200$, simulate Backward Euler and BDF2 again with $k = 5.10^{-2}$. Compute $\|\cdot\|_\infty = |\underline{u} - \underline{u}_{exact}|_\infty$ at $t = 1$. Redo the simulation with $k = 10^{-2}, 5.10^{-3}, 10^{-3}$. Plot the $\|\cdot\|_\infty$ at $t = 1$ vs k in log style. What is the slope of the error trend ?
5. Redo problem 1 and 2 with periodic boundary condition.