一维问题的有限差分求解

计算流体力学 Project1:part1~2

Professor 陈维建

BX2201913 包晨宇

2022 年 10 月 21 日

目录

1 作业要求以及实现方式

1.1 作业要求

用有限差分法 FDM 求解不同的一维方程问题。需要选取不同的初值条件、参数取值、方程形式与离散格式。其中各不同的输入条件选取如下

1. 不同的初值条件（包括三角型 Triangle 初始值与阶跃型 Step 初始值）；
2. 参数 C 选取不同取值：$0.1, 0.5, 1, 5, 10$；
3. 方程形式选取 Single Wave Equation，Heat Equation 和 Burgers Equations。
4. 离散格式包括时间层与空间层的：
    - 时间层有显示（explicit）与隐式（implicit）两种离散格式。
    - 空间层有向后（backward）欧拉、向前（forward）欧拉和中心（center）差分。

1.2 实现方式

计算流体力学（CFD）属于科学计算的范畴，其有数据量大、对计算资源要求高的特点，本次作业采用实现方式如下：

- 本次作业编程语言选取 julia[2]，以避免 C 等编译型语言工程性的厚重和 Python 等解释性语言的性能不足，适合用于作业中的 CFD 问题求解；
- 在时间层计算的隐式格式中，需要用到矩阵求逆，因为该矩阵往往是一个对角占优的稀疏矩阵，本次作业选用 SparseArrays.jl 库处理存放稀疏矩阵；
- 本次作业的计算数据文件以 CSV 格式存放；
- 本次作业的可视化工具选取 Plots.jl，以批量、动画化计算数据。
- 因本次作业公式图片较多，需进行索引，故采用 LaTex 撰写，部分示意图 tikz 生成。

2 有限差分格式的讨论

2.1 有限差分格式讨论

2.1.1 空间差分的讨论

某一变量 u 在 x 点处值为 $u(x)$，其在 $x+\Delta x$ 位置处用 Taylor 展开[1]，可以得到：

$$u(x+\Delta x)=u(x)+\Delta x\left(\frac{\partial u}{\partial x}\right)+\frac{1}{2}(\Delta x)^2\left(\frac{\partial^2 u}{\partial x^2}\right)+\circ(\Delta x^3) \tag{2.1}$$

利用这一方程，在考虑 Δx 本身趋近于 0 时，其高阶项可以忽略，虽然这会带来截断误差，但只要认为 Δx 本身尺度足够小，该近似是可以接受的。根据该方程，在考虑 $u(x)$ 处导数的时候，可以近似地用 $u(x)$ 和 $u(x+\Delta x)$ 有关 Δx 的线性组合表示：

$$\begin{aligned}\frac{\partial u}{\partial x}\Big|_{x=x} &\approx \frac{u(x+\Delta x)-u(x)}{\Delta x}\\ &=\frac{1}{\Delta x}u(x+\Delta x)+\left(-\frac{1}{\Delta x}\right)u(x)\end{aligned} \tag{2.2}$$

从另一个角度来看，上式其实是做了一个简单的事情：在仅仅知道 $u(x)$ 和 $u(x+\Delta x)$ 的情况下，在局部用一个线性函数对原函数进行了拟合：$u=kx+b$。记 $u(x)$ 在 x_0 处的值为 u_0，$u(x_0+\Delta x)$ 处的值为 u_1，记近似的函数符号为 $\tilde{u}(x)$，其在 $x\sim x+\Delta x$ 这一部分上，被含 u_0 和 u_1 参数的直线所近似表示：

$$\tilde{u}(x)=\frac{u_1-u_0}{\Delta x}(x-x_0)+u_0 \tag{2.3}$$

$\tilde{u}(x)$ 是在 $x_0\sim x_0+\Delta x$ 范围内对原函数 $u(x)$ 的一个不错的近似，因此其导数值也可以近似地认为是 $u(x)$ 的导数：

$$u'(x_0)\approx\tilde{u}'(x_0)=\frac{u_1-u_0}{\Delta x} \tag{2.4}$$

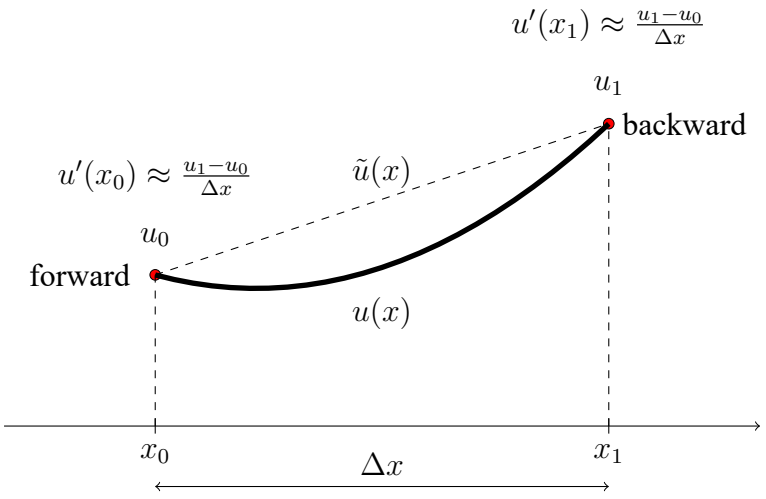


图 2.1: 向前与向后差分示意图

上式在 x_0 处认为 $u(x)$ 的导数为 $(u_1-u_0)/\Delta x$，这是对这条直线的左侧端点进行考量，如果认为该条直线的右侧端点导数值也是直线的斜率，如图 2.1即：

$$u'(x_1) \approx \tilde{u}'(x_1) = \frac{u_1-u_0}{\Delta x} \tag{2.5}$$

则也得到了右侧点的导数值。事实上，式 2.4表征的即为向前（forward）差分，式 2.5表征的即为向后（backward）差分。

于是很自然的，我们可能不满足于只靠两个点，用一条直线 $\tilde{u}(x)$ 来逼近原函数 $u(x)$，如果用更多的点，或许可以达到更高的精度。所以我们很自然地会把两个点拓宽到三个点。虽然在有限差分中一般都采用均匀的网格，但为了适应更一般的情形，我们假设有这样的三个点：$x=-\Delta_1, 0, \Delta_2$，其对应的函数 u 值分别记作 u_{-1}, u_0, u_1。三个点可以确定一条二次曲线来近似原函数：

$$\tilde{u}(x) = a_0 + a_1 x + a_2 x^2 = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \tag{2.6}$$

代入对应的三个点：$(-\Delta_1, u_{-1})$ $(0, u_0)$ (Δ_2, u_1) 得到如下线性方程组（用矩阵表达）：

$$\begin{bmatrix} 1 & -\Delta_1 & \Delta_1^2 \\ 1 & 0 & 0 \\ 1 & \Delta_2 & \Delta^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} u_{-1} \\ u_0 \\ u_1 \end{bmatrix} \tag{2.7}$$

于是根据式 2.6，可以将 $\tilde{u}(x)$ 写成：

$$\tilde{u}(x) = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} 1 & -\Delta_1 & \Delta_1^2 \\ 1 & 0 & 0 \\ 1 & \Delta_2 & \Delta_2^2 \end{bmatrix}^{-1} \begin{bmatrix} u_{-1} \\ u_0 \\ u_1 \end{bmatrix} \tag{2.8}$$

则 $u'(x)$ 可以用 $\tilde{x}'(x)$ 来近似表示如下：

$$\tilde{u}'(x) = \begin{bmatrix} 0 & 1 & 2x \end{bmatrix} \begin{bmatrix} 1 & -\Delta_1 & \Delta_1^2 \\ 1 & 0 & 0 \\ 1 & \Delta_2 & \Delta_2^2 \end{bmatrix}^{-1} \begin{bmatrix} u_{-1} \\ u_0 \\ u_1 \end{bmatrix} \tag{2.9}$$

上式是关于 x 的一次函数，这表明用三个点的值来近似表示这一段函数 $u(x)$ 时。其导数值可以用一条直线来表示（是富于变化的）。而这个表达是和网格选取 Δ、插值点取值 $u_i(i=-1,0,1)$ 有关的。

基于上式，现在考虑如下三个点处的导数值：

$$
\begin{aligned}
\tilde{u}'(-\Delta_1) &= \begin{bmatrix} -\frac{2\Delta_1+\Delta_2}{\Delta_1^2+\Delta_1\Delta_2} & \frac{\Delta_1+\Delta_2}{\Delta_1\Delta_2} & -\frac{\Delta_1}{\Delta_1\Delta_2+\Delta_2^2} \end{bmatrix} \begin{bmatrix} u_{-1} \\ u_0 \\ u_1 \end{bmatrix} \\
\tilde{u}'(0) &= \begin{bmatrix} -\frac{\Delta_2}{\Delta_1^2+\Delta_1\Delta_2} & \frac{\Delta_2-\Delta_1}{\Delta_1\Delta_2} & \frac{\Delta_1}{\Delta_1\Delta_2+\Delta_2^2} \end{bmatrix} \begin{bmatrix} u_{-1} \\ u_0 \\ u_1 \end{bmatrix} \\
\tilde{u}'(\Delta_2) &= \begin{bmatrix} \frac{\Delta_2}{\Delta_1^2+\Delta_1\Delta_2} & -\frac{\Delta_1+\Delta_2}{\Delta_1\Delta_2} & \frac{\Delta_1+2\Delta_2}{\Delta_1\Delta_2+\Delta_2^2} \end{bmatrix} \begin{bmatrix} u_{-1} \\ u_0 \\ u_1 \end{bmatrix}
\end{aligned} \tag{2.10}
$$

其中考虑一维网格是均匀的，有 $\Delta_1 = \Delta_2 = \Delta x$，那么 $\tilde{u}(0)$ 可以表示为：

$$
u(0) \approx= \tilde{u}'(0) = \frac{u_1 - u_{-1}}{2\Delta x} \tag{2.11}
$$

这也就是中心差分（central）格式。

2.1.2 对中心差分格式的一些延申

2.1.2.1 二阶导数

在式 2.9中，$\tilde{u}(x)$ 的二次导数可以求得，为：

$$
\begin{aligned}
\tilde{u}''(x) &= \begin{bmatrix} 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & -\Delta_1 & \Delta_1^2 \\ 1 & 0 & 0 \\ 1 & \Delta_2 & \Delta_2^2 \end{bmatrix}^{-1} \begin{bmatrix} u_{-1} \\ u_0 \\ u_1 \end{bmatrix} \\
&= \begin{bmatrix} \frac{2}{\Delta_1(\Delta_1+\Delta_2)} & -\frac{2}{\Delta_1\Delta_2} & \frac{2}{\Delta_2(\Delta_1+\Delta_2)} \end{bmatrix} \begin{bmatrix} u_{-1} \\ u_0 \\ u_1 \end{bmatrix} \\
&\downarrow \Delta_1 = \Delta_2 = \Delta x \\
&= \frac{u_{-1} + u_1 - 2u_0}{2\Delta x^2}
\end{aligned} \tag{2.12}
$$

上式表明，原函数 $u(x)$ 在借三个点用二次函数局部拟合的情况下，其二阶导数是可以表示的，且这三个点的二阶导数可以认为是上式 $\tilde{u}''$ 的值，因此若偏微分方程中出现 $\partial^2/\partial x^2$ 项，可以在网格局部选取三个点有关网格划分 Δx 的线性组合来表示。

2.1.2.2 壁面边界处的导数值

在对 PDE 进行求解时，网格一定会有边界处 x_0 与 x_N，在这两个地方显然不能用中心差分格式来进行导数值求取。但根据式 2.10中，我们可以把 x_0 看成位置在 $-\Delta_1$ 的点，得

到在 x_0 处的导数值：

$$\begin{aligned} u'(x_0) \approx \tilde{u}'(x_0) &= \begin{bmatrix} -\frac{2\Delta_1+\Delta_2}{\Delta_1^2+\Delta_1\Delta_2} & \frac{\Delta_1+\Delta_2}{\Delta_1\Delta_2} & -\frac{\Delta_1}{\Delta_1\Delta_2+\Delta_2^2} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} \\ &\downarrow \Delta_1 = \Delta_2 = \Delta x \\ &= \frac{1}{\Delta x}\left(-\frac{3}{2}u_0 + 2u_1 - \frac{1}{2}u_2\right) \end{aligned} \tag{2.13}$$

如图 2.2中表示左侧壁面 x_0 处导数值的差分近似，而最右侧 x_N 可以用类似的方法进行差分近似。

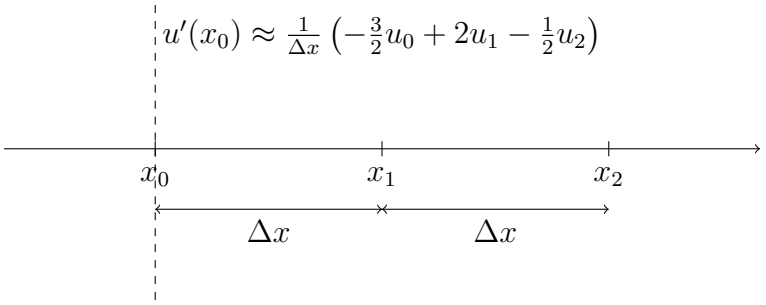


图 2.2: 壁面处的有限差分

2.1.2.3 如何看待这个差分过程

在网格划分确定、空间差分格式确定的情况下，在节点 i 处的导数近似值 u_i' 可以看成是节点值的一个线性组合：

$$u_i' = \sum_{k=0}^{N} a_{ik} u_k \tag{2.14}$$

其中 a_{ik} 取决于一维网格与差分格式，比如对于均匀网格的中心差分格式，有：

$$a_{ik} = \begin{cases} -\dfrac{1}{2\Delta x} & k = i-1 \\ \dfrac{1}{2\Delta x} & k = i+1 \\ 0 & \text{otherwise} \end{cases} \qquad (i \neq 0, N) \tag{2.15}$$

将 u_i' 组成的列向量记作 U'，u_i 组成的列向量记作 U，a_{ik} 组成的矩阵为 A（A 受制于网格 Δx 和差分格式 scheme），则上式可以写成一个矩阵乘法：

$$U' = A(\Delta x, \text{scheme})U \tag{2.16}$$

这里的 A 可以视作对一个离散点求导的差分算子，经 A 的作用，可以得到其对应点上的导数近似值。

显然在一维问题中，A 是一个对角占优的稀疏矩阵（sparse matrix），本次作业在求解中运用了 SparseArrays.jl 库对稀疏矩阵进行了列压缩存放，并用于每步计算中。

2.2 时间层离散的讨论

为方便起见，沿用在式 2.16中的表达：

$$U' = AU \tag{2.17}$$

并且以一维波动方程为例，进行时间层离散的讨论。

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \tag{2.18}$$

2.2.1 显式格式（explicit）

在考虑时间步长 Δt 的时候，可以将 $\partial U/\partial t$ 用 $n+1$ 时间层和 n 层的各个网格节点上的值 u_i 组成的列向量 U 表示：

$$\left(\frac{\partial U}{\partial t}\right)_n \approx \frac{U^{n+1} - U^n}{\Delta t} \tag{2.19}$$

结合波动方程表达式，用 AU^n 替换 $\partial u/\partial x$ 项，用 $(U^{n+1} - U^n)/\Delta t$ 项，可以得到离散情况下的时间推进格式：

$$U^{n+1} = (I - c\Delta t A)U^n \tag{2.20}$$

其中 I 为单位矩阵。这么做的好处是做法十分简单，在已知 U^n 的情况下，确定时间步长 Δt、空间差分格式与网（即 A），就能用一个简单的矩阵乘法求得下一个时间层的 U^{n+1}。但缺点是误差容易累积，导致计算结果发散。

值得注意的是，该计算中矩阵 $I - c\Delta t A$ 仍是一个大型稀疏矩阵。

2.2.2 隐式格式（implicit）

为了避免显式格式的计算误差，在波动方程中用 AU^{n+1} 替换 $\partial u/\partial x$ 项，就得到了隐式格式：

$$(I + c\Delta t A)U^{n+1} = U^n \tag{2.21}$$

为了得到 $n+1$ 层时间层的值，所需要进行的操作是对上式求逆：

$$U^{n+1} = (I + c\Delta t A)^{-1}U^n \tag{2.22}$$

虽然隐式格式稳定，但因为涉及到大型稀疏矩阵的求逆，可能在每一步时间计算上性能会比显式格式差很多。

2.3 编程中考虑的空间与时间差分格式的解耦

式 2.20与式 2.22是一维波动方程的显式与隐式格式（时间）。

本次作业为尝试性质的作业，因此需要对各种差分格式进行研究。对于每一个空间差分格式，需要写对应的显式与隐式两种程序。为了减轻工作量，注意到若网格确定、空间

差分格式确定，则式 2.20与式 2.22中的 A 是一致的。如完成某一空间差分格式的显式程序后，无需再重新书写隐式程序，只需将时间层的矩阵乘法改写成矩阵除法即可。

这表明时间差分格式与空间差分格式在程序中的体现可以解耦，对于不同的空间差分格式生成不同的 A，传给时间推进的函数，再选取时间差分格式，这样可以避免代码的冗余性，也方便维护修改。并且因为引用了 SparseArrays.jl 库，其对稀疏矩阵乘法的优化，不会让显式格式中的程序速度低于正常书写的循环程序。

2.4 差分格式的稳定性

以单波方程的不同差分形式为例，讨论差分格式的稳定性。其时间差分格式均选取为显示格式。

2.4.1 向后差分的稳定性分析

单波方程的的向后差分（显式格式）如下：

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}+c\frac{u_j^n-u_{j-1}^n}{\Delta x}=0 \tag{2.23}$$

其差分依赖区包含微分依赖区，且格式稳定。根据冯诺依曼稳定性分析，推导如下：

$$\begin{aligned} G&=1-c\frac{\Delta t}{\Delta x}\cdot(1-e^{-ik\Delta x})\\ \text{替换}\downarrow\ &c\frac{\Delta t}{\Delta x}=m\\ G&=1-m+m(\cos k\Delta x-i\sin k\Delta x)\\ G&=1-m(1-\cos k\Delta x)-im\sin k\Delta x\\ G&=a+ib \end{aligned} \tag{2.24}$$

其中 $a=1-m(1-\cos k\Delta x)$，$b=-m\sin k\Delta x$，于是有：

$$\left[\frac{a-(1-m)}{m}\right]^2+\left(\frac{b}{m}\right)^2=1 \tag{2.25}$$

上式表达的是圆心在 $(1-m,0)$ 处，半径为 $m(m>0)$ 的圆，为使得 $\sqrt{a^2+b^2}\le 1$，这就要求：

$$|1-m|+m\le 1 \tag{2.26}$$

即 $c\Delta t/\Delta x\le 1$。因此对于单波方程而言，其向后差分的显式格式在上述条件下是稳定的。

2.4.2 向前差分的稳定性分析

单波方程的向前差分（显式格式）如下：

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}+c\frac{u_{j+1}^n-u_j^n}{\Delta x}=0 \tag{2.27}$$

差分依赖区在左侧，微分依赖区在右侧；格式不稳定，推导如下，默认替换 $c\frac{\Delta t}{\Delta x}=m$：

$$\begin{aligned} G &= 1-m\cdot(e^{ik\Delta x}-1) \\ G &= 1+m-m\cos k\Delta x-im\sin k\Delta x \\ G &= a+ib \end{aligned} \tag{2.28}$$

类似地，将上述方程看成复平面上的一个圆：

$$\left[\frac{a-(1+m)}{m}\right]^2+\left(\frac{b}{m}\right)^2=1 \tag{2.29}$$

圆心在 $(1+m,0)$ 处，半径 $r=m$ 的圆，显然要求：

$$|1+m|+m\leq 1 \tag{2.30}$$

这样的 m 是不存在的，因此可以认定向前差分任意情况下都是不稳定的。

2.4.3 中心差分的稳定性分析

单波方程的中心差分（显式格式）如下：

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}+c\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x}=0 \tag{2.31}$$

差分依赖区包括微分依赖区，但是格式不稳定；推导如下，默认替换 $c\frac{\Delta t}{\Delta x}=m$：

$$\begin{aligned} G &= 1-m\cdot\frac{e^{ik\Delta x}-e^{-ik\Delta x}}{2} \\ G &= 1-im\sin k\Delta x \end{aligned} \tag{2.32}$$

为使增长因子 G 的模小于 1，则要求 $m=0$ 恒成立，这显然是不可能的。中心差分格式对于单波方程而言不稳定。因此虽然中心差分格式在计算中对空间的导数项逼近得更好，但不可以用于求解一维单波方程问题。

2.5 三个方程选取的差分形式

由于单波方程的差分形式已在前文中讨论过，已确定正确的差分形式为向后差分。因此本节不再讨论单波方程，而侧重于讨论 Heat Equation 与 Burgers Equation 的差分离散形式。

2.5.1 一维热传导方程的差分形式与稳定性讨论

一维热传导方程形式如下：

$$\frac{\partial u}{\partial t}=\alpha\frac{\partial^2 u}{\partial x^2}\quad \alpha>0 \tag{2.33}$$

如节 2.1.2.1中所述，空间上的二阶导数可以用中心差分的形式来表示：

$$\frac{\partial^2 u}{\partial x^2}|_{x=x_i} \approx \frac{u_{i+1}+u_{i-1}-2u_i}{\Delta x^2} \tag{2.34}$$

在该方程的边界处往往需要给定恒温壁条件（$u=\text{Const}$）或绝热壁（$\partial u/\partial x=\text{Const}$）等边界条件。

在该空间差分格式确定后，对于显式格式可以写成：

$$\frac{u_j^{n+1}-u_j^n}{\Delta t} = \frac{\alpha}{\Delta x^2}\left(u_{j+1}^n+u_{j-1}^n-2u_j^n\right) \tag{2.35}$$

考虑上式格式的稳定性，记 $m=\alpha\Delta t/\Delta x^2$，可以得到：

$$\begin{aligned} G-1 &= m\left(e^{ik\Delta x}+e^{-ik\Delta x}-2\right) \\ G &= 1-2m+2m\cos(k\Delta x) \\ G &= 1-4m\sin^2\frac{k\Delta x}{2} \end{aligned} \tag{2.36}$$

为使得 $|G|\le 1$，则要求 $m\le 1/2$ 恒成立，因此该格式的稳定性对 Δt 和 Δx 有一定要求。若考虑隐式格式，则可以将增长因子计算为：

$$\begin{aligned} 1-\frac{1}{G} &= m\left(e^{ik\Delta x}+e^{-ik\Delta x}-2\right) \\ G &= \frac{1}{1+4m\sin^2\frac{k\Delta x}{2}} \end{aligned} \tag{2.37}$$

上式在任何情况下满足增长因子的模小于 1，因此隐式格式无条件稳定。

另外，延续在单波方程中的编程思路，引入差分算子的记号，记二阶差分算子为 D_2，则可以将显式的中心差分格式写成：

$$U^{n+1}-U^n = \alpha\Delta t D_2 U^n \tag{2.38}$$

也就是：

$$U^{n+1} = (I+\alpha\Delta t D_2)U^n \tag{2.39}$$

因此其对应的隐式格式可以写成：

$$U^{n+1} = (I-\alpha\Delta t D_2)^{-1}U^n \tag{2.40}$$

所以在一维热传导的编程中，需要注意以下两点：

- 显式格式中需要控制时间步长，使得格式满足 CFL 条件；
- 边界处的边界条件需要正确给定（如等温壁或绝热壁）；
- 在边界条件给定的时候，可能会破坏形如式 2.20或 2.22中矩阵乘法的格式，这需要另作处理。

2.5.2 一维热 Burgers 方程的差分形式与稳定性讨论

2.5.2.1 显式格式的守恒形式与非守恒形式

一维 Burgers 方程的形式如下：

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}=\nu\frac{\partial^2 u}{\partial x^2} \tag{2.41}$$

根据文献[3]，对时间层离散用显式格式，对 $\partial u/\partial x$ 用中心差分格式离散，对 $\partial^2 u/\partial x^2$ 也用中心差分格式离散如下：

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}+u_j^n\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x}=\nu\frac{u_{j+1}^n+u_{j-1}^n-2u_j^n}{\Delta x^2} \tag{2.42}$$

该参考文献指出，此差分格式在初始误差 ξ_j^0 的模小于 1，且 $\nu\Delta t/\Delta x^2<1/2$ 时，格式稳定。

另外，注意到：

$$u\frac{\partial u}{\partial x}=\frac{1}{2}\frac{\partial u^2}{\partial x} \tag{2.43}$$

因此可以有守恒形式差分格式如下：

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}+\frac{(u_{j+1}^n)^2-(u_{j-1}^n)^2}{4\Delta x}=\nu\frac{u_{j+1}^n+u_{j-1}^n-2u_j^n}{\Delta x^2} \tag{2.44}$$

同样引入一阶差分算子 D_1 和二阶差分算子 D_2，并记 $\mathrm{diag}(V)$ 表示将向量 V 张成一个对角矩阵，且其对角元素为向量各元素，则可以将非守恒差分格式写成：

$$U^{n+1}-U^n=\Delta t\left[-\mathrm{diag}(U^n)D_1+\nu D_2\right]U^n \tag{2.45}$$

也就是非守恒形式的显式中心差分为：

$$U^{n+1}=\left[I-\Delta t\mathrm{diag}(U^n)D_1+\nu\Delta t D_2\right]U^n \tag{2.46}$$

而守恒形式的显式差分格式可以写成：

$$U^{n+1}=-\frac{\Delta t}{2}D_1(U^n)^2+(I+\nu\Delta t D_2)U^n \tag{2.47}$$

如果不采用中心差分，比如对 uu_x 项用向后差分，只需要修正 D_1 为向后差分算子。

2.5.2.2 半隐式格式

对于隐式格式，式 2.46应当写成：

$$\left[I+\Delta t\mathrm{diag}(U^{n+1})D_1-\nu\Delta t D_2\right]U^{n+1}=U^n \tag{2.48}$$

显然上式是无法直接求逆的，因为第 $n+1$ 时间层的 U^{n+1} 值未知，但在实践操作中，每一步 U^n 变化量很小，可以近似地用 $\mathrm{diag}(U^n)$ 项取代 $\mathrm{diag}(U^{n+1})$ 得到：

$$U^{n+1}=\left(I+\Delta t\mathrm{diag}(U^n)A_1-\nu\Delta t A_2\right)^{-1}U^n \tag{2.49}$$

上式对应的迭代格式为半隐式迭代格式：

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}+u_j^n\frac{u_{j+1}^{n+1}-u_{j-1}^{n+1}}{2\Delta x}=\nu\frac{u_{j+1}^{n+1}+u_{j-1}^{n+1}-2u_j^{n+1}}{\Delta x^2} \tag{2.50}$$

当然，在引入边界条件时，需要考虑这个矩阵乘法作局部的修正。

3 程序设计的框架与编写思路

3.1 程序框架

在 CODE 文件夹下，有 src，test，data，example，image，doc，draft 共七个文件夹。其各含有的文件（夹）如下：

1. src：源代码文件夹，包含四个文件如下：
    - Base1D.jl：一维计算公用程序，里面包含了一维时空网格 Grid 结构体，一阶的向前、向后、中心差分算子（为稀疏矩阵）生成函数；
    - SingleWave1D.jl：一维单波方程求解程序，引入了 Base1D.jl，考虑了三种空间差分格式和两种时间计算格式构造了求解器；
    - Heat1D.jl：一维热传导方程求解程序，引入了 Base1D.jl，考虑了两种边界条件和两种时间格式，空间差分用中心差分构造的求解器；
    - Burgers1D.jl：一维 Burgers 方程求解程序，引入了 Base1D.jl，考虑了一种边界条件和两种时间格式，空间差分用中心差分构造的求解器。
2. test：测试程序文件夹，包含三个文件 SingleWave1D_test.jl，Heat1D_test.jl 和 Burgers1D_test.jl 文件，对三类方程的源代码进行了实例化测试；
3. data：包含 SingleWave，Heat 和 Burgers 三个文件夹，各文件夹下对应不同情形的测试程序；因为该部分文件大小较大，因此可能不会打包上交——但这个文件夹可以用 example 中的程序生成；
4. example：包含 visualize.ipynb 绘图 jupter 笔记本，以及 SingleWave，Heat 和 Burgers 三个文件夹，各文件夹下对不同参数取值、差分格式、初始条件、边界条件进行了计算，需要注意的是除了 jl 代码源文件外，还有两个 sh 文件：
    - run_all.sh：LINUX 系统下的运行脚本，会调用该目录下所有计算 jl 文件，生成数据文件保存在 data 目录相应文件夹下；
    - animate_all.sh：LINUX 系统下的运行脚本，调用该目录下所有可视化 jl 文件，生成 gif 动画保存在 image 目录相应文件夹下。
5. image：包含 SingleWave，Heat 和 Burgers 三个文件夹，各文件夹下对应不同情形的测试程序；因为该部分文件大小较大，因此可能不会打包上交——但这个文件夹可以用 example 中的程序生成；
6. doc：为程序编写过程中的一些记录以及公式推导，包括 markdown 文件和 mathematica 笔记本，以及一个 framework.txt，是用 tree 命令生成的当前目录所有文件；
7. draft：为程序开发中的一些草稿，可能不会打包上传，可忽略。

3.2 编写思路

因为在编程过程中，随着编写进展，发现有很多重复的功能需要编写，于是将三个方程求解过程中都需要的东西放在 Base1D.jl 文件中，作为一维问题的公用函数库。

虽然一维问题可以用简单的均匀网格和均匀时间步长来处理，无非是对网格进行暴力的加密。但本程序考虑到后续可能的修改与添加功能，考虑了网格的不均匀情况，因此会有式 2.10中的推导，所以本次作业中的一维网格统一用 Base.Grid 结构体管理。

另外在 Base.jl 中有根据一维网格 Grid 生成一阶差分算子 D_1（向前、向后和中心）和二阶差分算子 D_2（中心）的函数（返回值为 SparseMatrix）。差分算子的含义在第二节中已叙述过，其可以避免在时间层显式与隐式情况下还要再分开生成差分矩阵的过程。

在每一个方程对应的 solve 函数中，会根据给定的时间差分格式 scheme_t 和空间差分格式 scheme_x 选取对应的算法，对于边界条件也会按照给定的算法计算。默认情况下选用隐式格式的中心差分格式，边界条件选取为第一类边界条件。

该程序设计的好处在于可以较方便的加入新的方程形式、加入新的差分格式以及添加新的边界条件类型。提高代码复用率，方便维护，且可以较自由地定义算例。

4 一维 SingleWave 方程的有限差分解

本次作业的空间网格选取为 $[-10, 10]$ 区间上，以 $\Delta x = 0.1$ 为步长取的均匀网格。空间上 Δt 选取为 $0.005s$，计算截至为 $5s$。下面对一维 SingleWave 的有限差分解展开讨论。

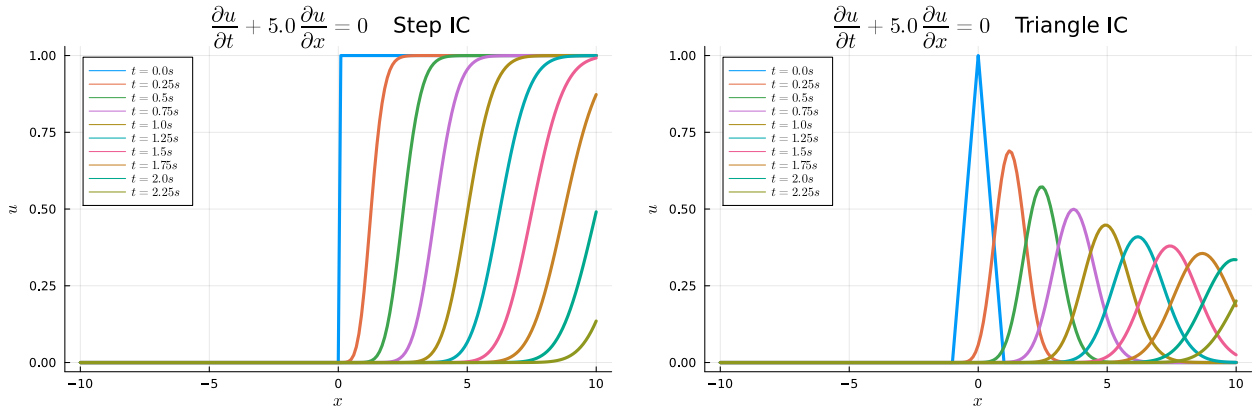
4.1 SingleWave 方程特性研究

4.1.1 方程的时间演化特性

在节 2.4.1中讨论过，对于单波方程而言其向后差分的解是稳定的，只是在显式格式里要求离散格式满足 CFL 条件，即 $c\Delta t/\Delta x \leq 1/2$，而隐式格式无条件稳定。

因为参数 c 的取值最大到 10，在前述的 Δx 和 Δt 的取值下恰好能满足 CFL 条件。为了保险起见，我们用隐式格式、向后差分的离散方法，来研究一维单波方程的特性。

取 $c = 5.0$，考虑在阶跃（Step）初始条件下方程的时间演化特性如下：



(a) Step IC. (b) Triangle IC.

图 4.1: 单波方程的时间演化特性

如图 4.1a所示，对于 $c = 5.0$ 的情形，在时间 t 每隔 0.25 秒的情况下，整个波形从最初的 $t = 0s$ 开始向右侧移动。且注意到 $t = 1.0s$ 时，整个波形中心对称点恰好移动到 $x = 5.0$ 的位置上，另外在等间隔时间内，波形向右侧移动位移几乎一致，因此可以认为整个波形在以 $c = 5.0$ 的速度在整体向右侧运动。

另外，若取三角型（Triangle）初始条件，可以得到图像。通过图 4.1b可以发现，单波方程还有一个特性是会“抹平”波形。在图 4.1a中，图形从最初尖锐的阶跃信号变成类似 Logistic 形式的光滑的形式，者可以看成是系统对突跃形的初始条件做出的连续化。而在图 4.1b中，整个波形在以 $c = 5.0$ 的速度向右侧传播过程中，波形从尖锐的三角信号被逐渐“抹平”为正态分布型的波形。

因此总结单波方程的时间演化特性如下：

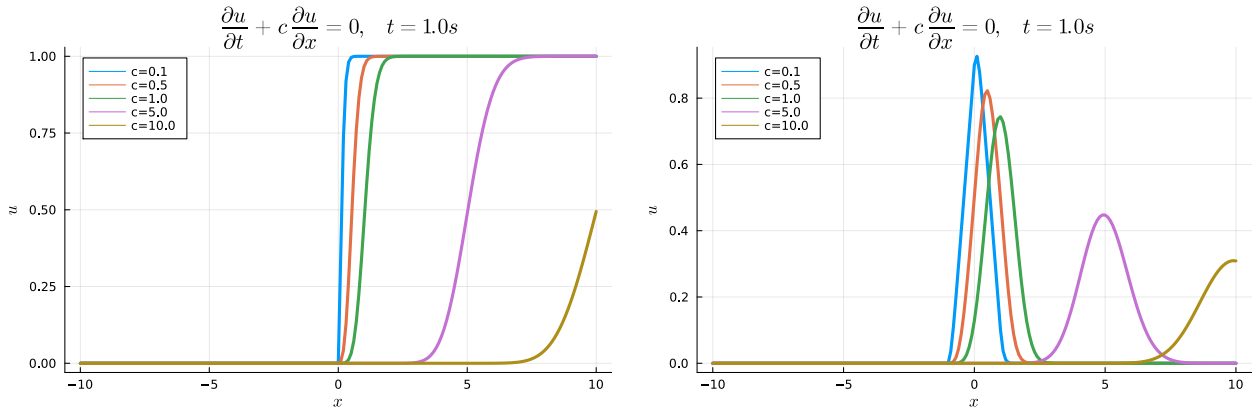
- 整个波形以 c 的速度沿 x 轴正方向传播；
- 波形会由尖锐的形状变得平滑。

4.1.2 参数 c 的作用

对于参数 c，其在前节中的作用已从时间演化的角度验证为向右侧传播的速度。因此感性认识而言，对于不同的 c，在同一个时刻，其传播的距离不一样。

用向后差分、隐式格式的方法，考察 $c = 0.1, 0.5, 1.0, 5.0, 10.0$ 情况下，时间 $t = 1.0s$ 的传播情况，如图 4.2a中展示的是单波方程在阶跃初始条件下参数取值的影响，图 4.2b为三角初始条件的。

可见 c 对于波形的影响就是其向右侧 x 轴正方向传播速率。在传播同等时间的情形下，c 越大，传播的距离越长，对初始波形的平滑作用程度越高。

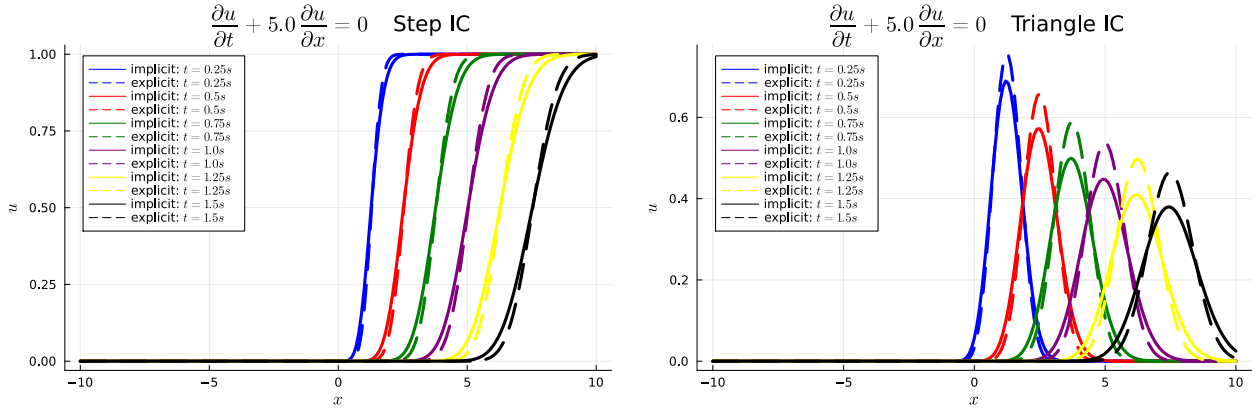


(a) Step IC. (b) Triangle IC.

图 4.2: 单波方程参数 c 取值的影响

4.2 显式格式与隐式格式的比较

为比较显式与隐式解法，这里选取 $c = 5.0$ 的空间向后差分格式进行考察，如图 4.3。



(a) Step IC. (b) Triangle IC.

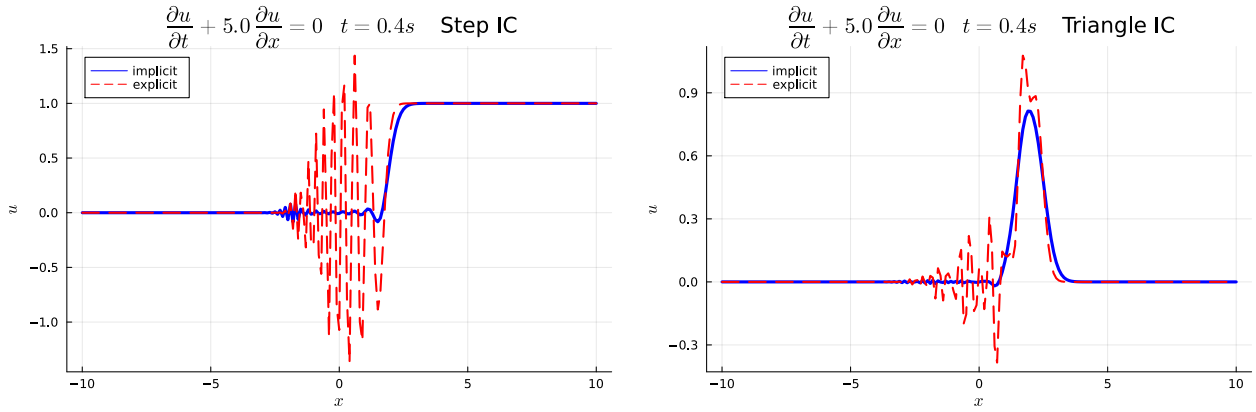
图 4.3: 单波方程显式格式与隐式格式的比较

该图中显式格式用虚线表示，隐式格式用实线表示。总体来看隐式格式导致的波形更加“矮胖”，这表明，在选取隐式格式时，单波方程“抹平”波形的效应更强，整体获得的数值解更加平滑；而相对而言，显式格式获得的解的“抹平效应”并没有那么强，整体获得的数值解更加“尖锐”。

4.3 中心差分与向前差分有限差分解的不稳定性

4.3.1 中心差分格式的不稳定性

这里先考虑空间上的中心差分格式，并且选择 $c = 5.0$，$t = 0.4s$ 的时候，在阶跃和三角初始条件下的波形。如图 4.4中展示了两种不同初始条件，两种时间格式下的波形图。



(a) Step IC. (b) Triangle IC.

图 4.4: 单波方程中心差分格式的不稳定性

图 4.4a中表明，阶跃型初始条件下，用中心差分格式会导致计算结果不稳定。在时间离散上用显式格式时（红色虚线），可以发现在本不应该有扰动的 x 负半轴产生了振荡，这是由错误的差分格式带来的。即使是采用理应更稳定的隐式格式（蓝色实线），也可以很明显地发现有一束“噪音”在 $x = -2$ 的位置，这是从右侧靠中心差分格式传来的。同样的，在图 4.4b中也能得到相似的结论。

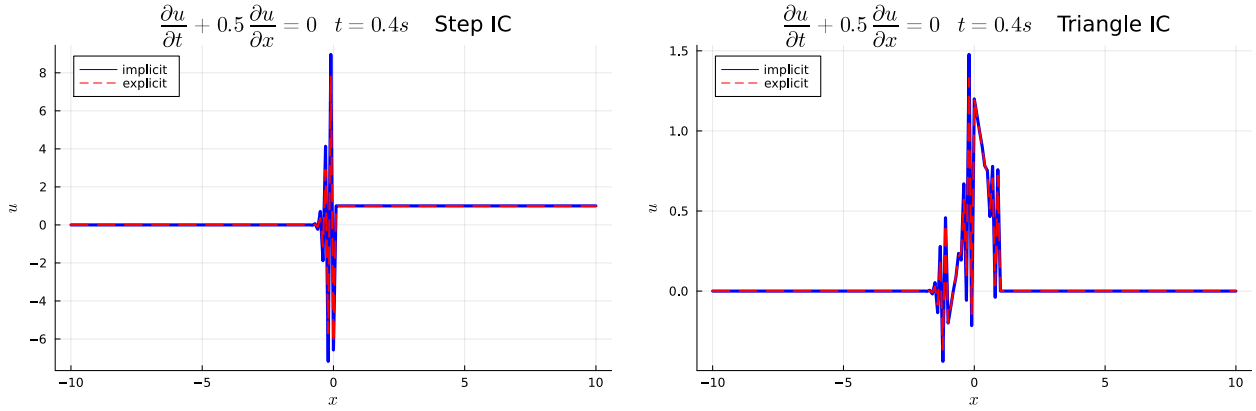
虽然中心差分格式不能得到正确的解，但图 4.4可以表明，在空间差分格式一致时，隐式格式（implicit）还是会比显式格式（explicit）有更好的数值稳定性。

4.3.2 向前差分格式的不稳定性

因为向前差分发散的很快，因此这里取 $c = 0.5$，$t = 0.4s$ 时的波形图，同样在阶跃和三角初始条件下的波形如图 4.5中所示。

不难发现即使在 c 取值很小时（是前文中中心差分格式 c 取值的 $1/10$），其格式不稳定性在 $t = 0.4s$ 时已经很显著了。

对于阶跃型初始条件，其在 $x = 0$ 处发生了较为剧烈的振荡；同样的，在三角型初始条件下，在 $x = \pm 1$ 处也有很明显的振荡。这都是由方程的错误离散格式导致的。



(a) Step IC. (b) Triangle IC.

图 4.5: 单波方程向前差分格式的不稳定性

4.4 小结

经过本章节对一维 SingleWave 单波方程有限差分解的讨论，可以得到以下经验和结论：

1. 单波方程的特性是将波形沿着 x 轴正方向传到，其有指向性；且参数 c 决定了该传播速度的快慢；该方程还会一定程度上抹平波形。

2. 单波方程仅在空间是采取向后差分的时候才是稳定且相容的；其中采取显式格式时需要满足 $c\Delta t/\Delta x \leq 1/2$。

5 一维 Heat 方程的有限差分解

本次作业的空间网格选取为 $[-10, 10]$ 区间上，以 $\Delta x = 0.1$ 为步长取的均匀网格。空间上 Δt 选取为 $0.005s$，计算截至为 $5s$。下面对一维 Heat 方程的有限差分解展开讨论。

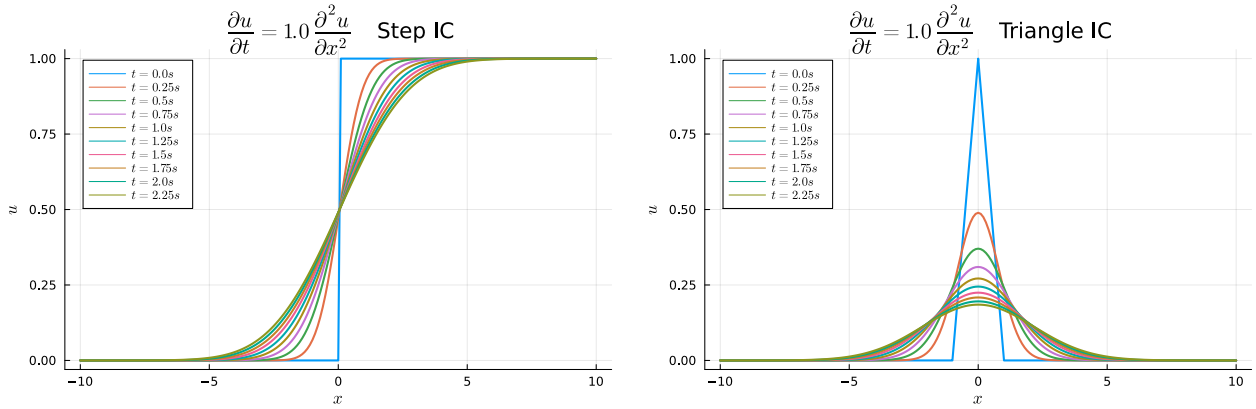
5.1 Heat 方程特性研究

5.1.1 方程的演化特性

在研究方程演化特性时，我们固定采用中心差分、隐式解，并且给定第一类边界条件：

$$u(-10, t) = \text{Const} = 0 \tag{5.1}$$

考察 Heat 方程的时间演化特性时，我们选取参数 $\alpha = 1.0$ 时，各个时刻的波形图。在图 5.1中为两种不同初始条件下的 Heat 方程不同时刻的波形分布。



(a) Step IC. (b) Traingle IC.

图 5.1: Heat 方程的时间演化特性

如图 5.1a中，在阶跃型初始条件下，$\alpha = 1.0$ 在时刻 $0, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25s$ 时的波形。随着时间增长，该阶跃扰动同时向 x 的正负半轴同时传播（扩散），并且因为微分方程解平滑性要求，会产生“抹平效应”，将尖锐的阶跃初始信号抹平为平滑的类 Logistic 型波形。

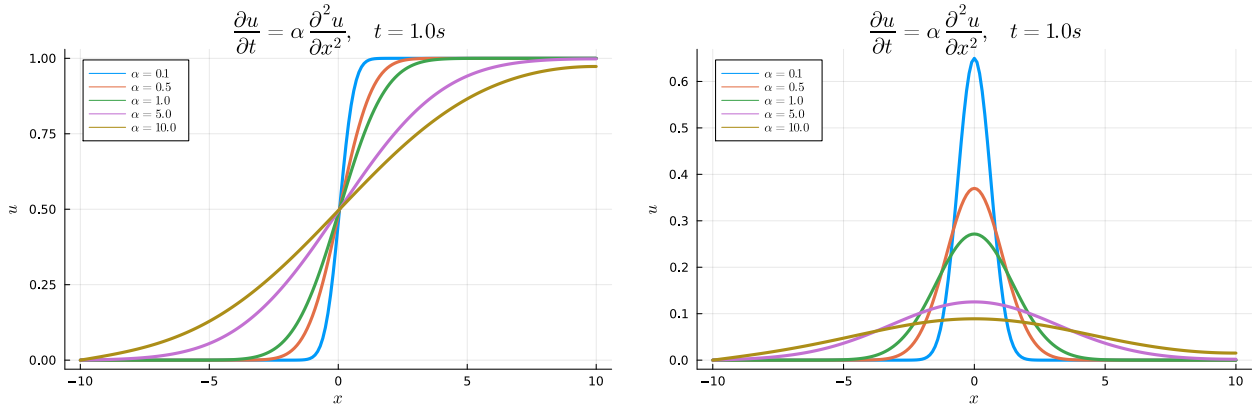
在图 5.1b中，同样的在三角形初始条件情形下，可以发现除了波形向两侧“扩散作用”，整个尖锐的三角部分“垮塌下来”，以满足微分方程的平滑性要求。

所以总结一维 Heat 方程的时间演化特性如下：

- 整个波形以某一速度同时向 x 正负半轴两侧扩散；
- 波形会由尖锐形状变得平滑。

5.1.2 参数 α 的作用

在这里仍然固定选取中心差分格式的隐式解下的一维 Heat 波形，边界条件设定同前。考虑 $t = 1.0s$ 时刻剖面上，参数 α 的不同取值对波形的影响，如图 5.2所示。



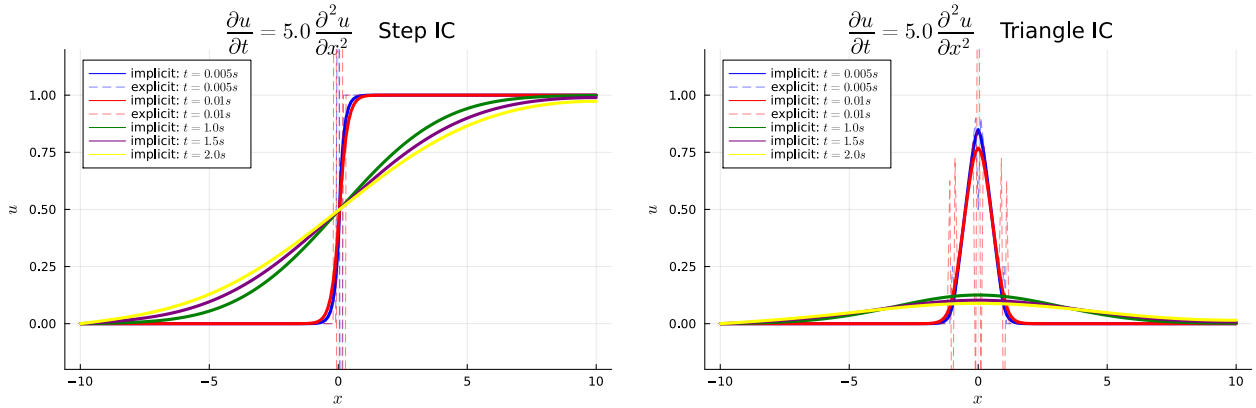
(a) Step IC. (b) Traingle IC.

图 5.2: Heat 方程参数 α 的影响

图 5.2a表示的是阶跃型初始条件下不同 α 取值的方程波形，图 5.2b表示的是三角型初始条件下不同 α 取值的方程波形。由此可见，在差分格式一定时，方程传播固定时长下，参数 α 取值越大，波形传播扩散得越快，波形被“抹平”的趋势越明显。

5.2 显式格式与隐式格式的比较

计算中要求 $\alpha\Delta t/\Delta x^2 \leq 1/2$，在本次作业中 $\Delta t = 0.005s$，$\Delta x = 1.0s$ 时，在 $\alpha = 1.0$ 时，显式格式仍然稳定。本次作业中在 $\alpha = 1.0$ 的显式隐式计算均收敛，因此不在报告中作展示。



(a) Step IC. (b) Traingle IC.

图 5.3: Heat 方程显式与隐式格式比较

为了体现显式格式与隐式格式的差别，本节考察在第一类边界条件（如式 6.1）$\alpha = 5.0$ 时的显式与隐式波形。因为在可视化过程中，发现此时显式格式发散得特别快，因此显式格式只取前两步可视化示意，隐式格式取较长演化时间展示，如图 5.3。

图 5.3a为初始条件为阶跃情形下显式隐式格式计算结果比较，图 5.3b为初始条件为三角情形下显式隐式格式计算结果比较。尽管显式格式只绘制了两步时间步长的波形结果，但可以明显发现计算值已经发散。而相对的，隐式格式在 α 很大，在计算多步的情况下仍然能保持相当的稳定性。因为 $\alpha = 10.0$ 时的隐式计算结果收敛，而显式计算结果发散极快，无法在同一张图中绘制作比较，因此本节仅可视化 $\alpha = 5.0$ 的情形波形对比。

上述的讨论中表明，Heat 方程的显式格式在满足 CFL 条件下，可以保证计算结果收敛。但一旦时空网格不满足 CFL 条件，显式格式很容易发散。而隐式格式则无条件收敛。

5.3 第一类边界条件的处理与结果

在处理热扩散 Heat 方程时，需要给定边界条件，而边界条件一般分为第一类边界条件和第二类边界条件。本次作业的程序设计中考虑了相应的函数接口，并在程序内部对不同边界条件做了处理。先考虑第一类边界条件的处理方法与结果（当前仅在 x 左端处设置边界条件）。

5.3.1 显式格式中第一类边界条件的处理

对于第一类边界条件，可以认为其在边界 $x = x_0$ 处给定某一函数 $f(t)$，使：

$$u(x_0, t) = f(t) \tag{5.2}$$

因此在 t_n 时间层的边界处 $u(x_0, t_n) = f(t_n)$ 是给定的。考虑式 2.39中 Heat 方程的显式中心差分格式：

$$U^{n+1} = AU^n \quad (A = I + \alpha \Delta t D_2) \tag{5.3}$$

因为 $u_1^{n+1} = f(t_{n+1})$ 已知，且 u_j^n 均已知（假设网格节点下标从 1 开始），所以在计算中无需考虑矩阵乘法中 A 的第一行乘 U^n 列向量，而只需直接赋值 $f(t^{n+1})$ 给 u_1^{n+1}。

5.3.2 隐式格式中第一类边界条件的处理

隐式格式中，处理方法要比显式格式稍微复杂一点。在式 2.40中，Heat 方程的隐式差分格式为：

$$BU^{n+1} = U^n \quad (B = I - \alpha \Delta t D_2) \tag{5.4}$$

其中 U^n 已知，且 $u_1^{n+1} = f(t^{n+1})$ 已知（当前仅在一端设置边界条件），考虑网格点共 M 个，方程可以改写为：

$$\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1M} \\ b_{21} & b_{22} & \cdots & b_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ b_{M1} & b_{M2} & \cdots & b_{MM} \end{bmatrix} \begin{bmatrix} f(t^{n+1}) \\ u_2^{n+1} \\ \vdots \\ u_M^{n+1} \end{bmatrix} = \begin{bmatrix} u_1^n \\ u_2^n \\ \vdots \\ u_M^n \end{bmatrix} \tag{5.5}$$

将已知的 $f(t^{n+1})$ 移动到方程右端去，能够得到 $u_j^{n+1}(j=2\sim M)$ 的解：

$$\begin{bmatrix} u_2^{n+1} \\ \vdots \\ u_M^{n+1} \end{bmatrix} = \begin{bmatrix} b_{22} & \cdots & b_{2M} \\ \vdots & \ddots & \vdots \\ b_{M2} & \cdots & b_{MM} \end{bmatrix}^{-1} \left(\begin{bmatrix} u_2^n \\ \vdots \\ u_M^n \end{bmatrix} - f(t^{n+1}) \begin{bmatrix} b_{21} \\ \vdots \\ b_{M1} \end{bmatrix} \right) \tag{5.6}$$

5.3.3 第一类边界条件的程序内设置

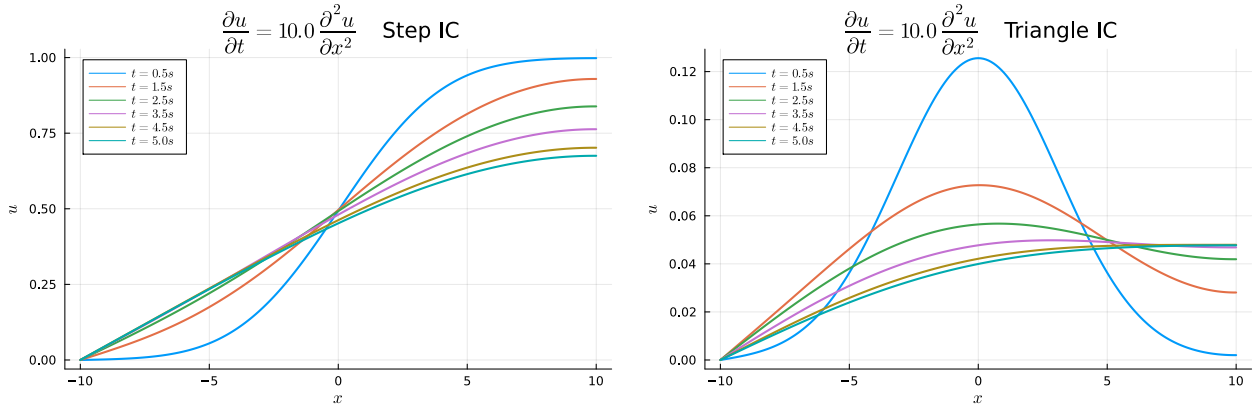
为了方便使用者更自由地定义边界条件，本程序的边界条件设置允许用户将边界条件 $f(t)$ 以函数句柄新式传入。也就是不仅支持恒温壁（第一类边界条件的一个特例），还支持时变的第一类边界条件输入。

5.3.4 第一类边界条件的计算结果

因为在 α 较小时，在计算的 $5s$ 内波形未传播到边界 $x_0=-10$ 的位置处，难以反映边界条件施加的影响。而 α 较大时，显式格式会发散。为了更加清晰地反映第一类边界条件施加的影响，本节给定 $\alpha=10.0$ 的中心差分、隐式格式计算结果，展示波形在不同时间层上的形状图比较，如图 5.4所示。

其中右侧不给定边界条件，左侧的边界条件给定为：

$$u(x_0,t)=u(-10,t)=f(t)=0 \tag{5.7}$$



(a) Step IC. (b) Traingle IC.

图 5.4: Heat 方程第一类边界条件施加影响

图 5.4a中展示的是在阶跃型初始条件下，左侧施加第一类边界条件情况下的时间演化波形图，可以很明显地发现左端由恒定值控制下，整个波形像风筝一样被“牵住了”，从而在边界处会有限制回传，控制住整个波形的位置；而另一端自由端则随传播逐渐“垮塌下来”。图 5.4b则展示的是在三角型初始条件下，左侧施加第一类边界条件情况下的时间演化波形图。同样的可以发现整个波形受左端边界条件牵制，逐步向下“垮塌”平滑化。

5.4 第二类边界条件的处理与结果

5.4.1 显式格式中第二类边界条件的处理

对于第二类边界条件，可以认为其在边界处 $x = x_0$ 处的梯度值给定位某一函数：

$$\frac{\partial u}{\partial x}(x_0, t) = u_x(x_0, t) = f(t) \tag{5.8}$$

记 $u^n_{x,1}$ 表示第 n 时间层上，边界点处的梯度值：

$$u^n_{x,1} = f(t_n) \tag{5.9}$$

本作业中在对第二类处理方式是这样的（这是我自己推导的，可能有不对的地方，希望老师指正）。根据式 2.10中的推导，我们可以用 $u^n_1 \sim u^n_4$ 这四个值，结合网格信息，计算得到在 $2 \sim 3$ 节点上的一阶导数值（记 h_k 表示 x_k 到 x_{k+1} 的长度）：

$$\begin{bmatrix} u^n_{x,2} \\ u^n_{x,3} \end{bmatrix} = \begin{bmatrix} -\frac{h_2}{h_1^2+h_1h_2} & \frac{h_2-h_1}{h_2h_1} & \frac{h_1}{h_1h_2+h_2^2} & 0 \\ 0 & -\frac{h_3}{h_2^2+h_2h_3} & \frac{h_3-h_2}{h_3h_2} & \frac{h_2}{h_2h_3+h_3^2} \end{bmatrix} \begin{bmatrix} u^n_1 \\ u^n_2 \\ u^n_3 \\ u^n_4 \end{bmatrix} \tag{5.10}$$

此时 $u^n_{x,i}(i = 1, 2, 3)$ 已经获知，再使用一次式 2.10中的推导，取 $-\Delta_1$ 处的导数值，既可以得到最终边界处的二阶导数值 $u^n_{xx,1}$：

$$u^n_{xx,1} = \begin{bmatrix} -\frac{2h_1+h_2}{h_1^2+h_1h_2} \\ \frac{h_1+h_2}{h_1h_2} \\ -\frac{h_1}{h_1h_2+h_2^2} \end{bmatrix}^{\mathrm{T}} \left(\begin{bmatrix} f(t_n) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\frac{h_2}{h_1^2+h_1h_2} & \frac{h_2-h_1}{h_2h_1} & \frac{h_1}{h_1h_2+h_2^2} & 0 \\ 0 & -\frac{h_3}{h_2^2+h_2h_3} & \frac{h_3-h_2}{h_3h_2} & \frac{h_2}{h_2h_3+h_3^2} \end{bmatrix} \begin{bmatrix} u^n_1 \\ u^n_2 \\ u^n_3 \\ u^n_4 \end{bmatrix} \right) \tag{5.11}$$

将其展开，可以得到 $u^n_{xx,1}$ 为 $u^n_j(j = 1 \sim 4)$ 的一个线性组合再叠加上某一个常数：

$$u^n_{xx,1} = \sum_{j=1}^{4} \hat{d}^2_{1j} u^n_j - \frac{2h_1 + h_2}{h_1^2 + h_1h_2} f(t_n) \tag{5.12}$$

将差分算子 D_2 第一行中的前 4 个元素 d^2_{1j} 替换为上式中的 $\hat{d}^2_{1j}$，得到修正后的二阶差分算子 $\hat{D}_2$，于是二阶差分（用 U^n_{xx} 表示 U^n 的二阶导数值）可以从最原始的 $U^n_{xx} = D_2U^n$ 改写为如下引入第二类边界条件的形式：

$$U^n_{xx} = \hat{D}_2U^n + f^n \tag{5.13}$$

其中 f_n 表示如下：

$$f^n = -\frac{2h_1 + h_2}{h_1^2 + h_1h_2} \begin{bmatrix} f(t_n) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.14}$$

于是引入第二类边界条件的显式格式完整形式可以写成如下的格式:

$$U^{n+1} = (I + \alpha\Delta t\hat{D}_2)U^n + \alpha\Delta t f^n \tag{5.15}$$

5.4.2 隐式格式中第二类边界条件的处理

隐式格式的处理逻辑和显式格式类似，只是将式 5.11改写成第 $n+1$ 时间层的。因为修正后的二阶差分算子 $\hat{D}_2$ 只与网格有关，因此可以直接写作：

$$U_{xx}^{n+1} = \hat{D}_2 U^{n+1} + f^{n+1} \tag{5.16}$$

于是隐式格式可以整体写作如下形式：

$$(I - \alpha\Delta t\hat{D}_2)U^{n+1} = U^n + \alpha\Delta t f^{n+1} \tag{5.17}$$

因为 f^{n+1} 已知，所以要求 U^{n+1} 只需简单的求逆就能得到下一个时间层的 U^{n+1}：

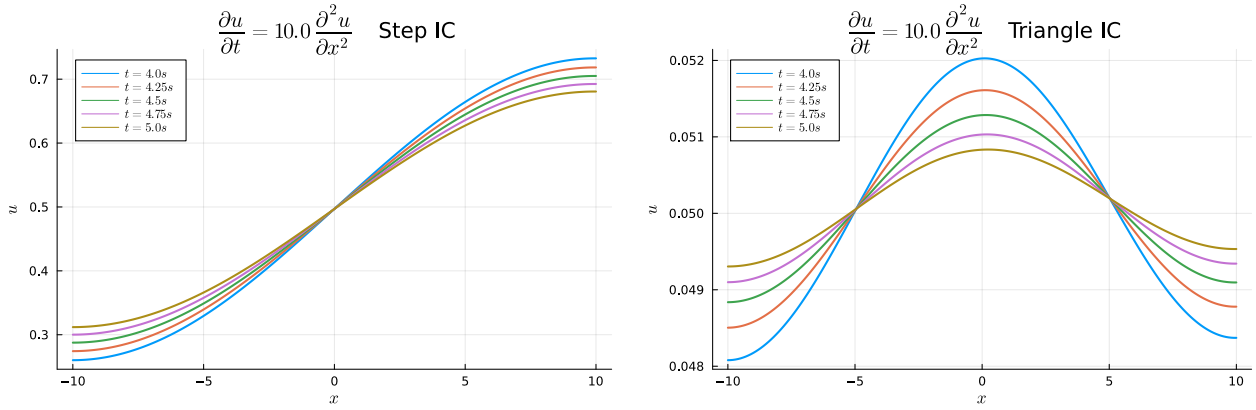
$$U^{n+1} = (I - \alpha\Delta t\hat{D}_2)^{-1}\left(U^n + \alpha\Delta t f^{n+1}\right) \tag{5.18}$$

5.4.3 第二类边界条件的计算结果

与第一类边界条件施加时一致，在 α 较小时，在计算的 $5s$ 内波形未传播到边界 $x_0 = -10$ 的位置处，难以反映边界条件施加的影响。而 α 较大时，显式格式会发散。为了更加清晰地反映第二类边界条件施加的影响，本节给定 $\alpha = 10.0$ 的中心差分、隐式格式计算结果，展示波形在不同时间层上的形状图比较，如图 5.5所示。

其中右侧不给顶边界条件，左侧的边界条件给定为：

$$u_x(x_0, t) = u_x(-10, t) = f(t) = 0 \tag{5.19}$$



(a) Step IC.

(b) Traingle IC.

图 5.5: Heat 方程第二类边界条件施加影响

图 5.5a 展现的是阶跃初始条件下施加第二类边界条件的波形图，图 5.5b 展现的是三角初始条件下施加第二类边界条件的波形图。此时虽然左侧边界上的 u 值有了偏移变化，但是其沿 x 轴的导数值始终恒定为 0，整个波形的左侧像被一双大手水平“按住”了。

图 5.5b中可以从 $t = 4.0s$ 蓝色波形上发现，没有施加第二类边界条件的右侧波形“上翘”幅度比施加了第二类边界条件的左侧更明显。综上所述，前述对第二类边界条件的处理从结果来看是合理的（希望老师给出指点，能否作如上的处理）。

5.5 小结

经过本章节对于一维 Heat 方程的有限差分解的讨论，可以得到如下经验和结论：

1. 一维 Heat 方程的效果是向 x 轴两侧扩散，且参数 α 值越大，扩散越快；方程本身也有均匀性的能力，会一定程度上抹平波形。

2. 中心差分格式对于 Heat 方程是自然而可行的差分形式；使用显式格式时需要满足 CFL 条件，而隐式格式则是无条件稳定的。

3. 第一类边界条件限制一端的取值，若为常数可以视作“等温壁”条件；第二类边界条件给定一端的空间梯度值，若为 0 可以看作“绝热壁”条件。这两个边界条件的引入都会破坏迭代形式，需要对原始的时间推进格式在边界处作特殊处理。

6 一维 Burgers 方程的有限差分解

本次作业的空间网格选取为 $[-10, 10]$ 区间上，以 $\Delta x = 0.1$ 为步长取的均匀网格。空间上 Δt 选取为 $0.005s$，计算截至为 $5s$。下面对一维 Burgers 方程的有限差分解展开讨论。

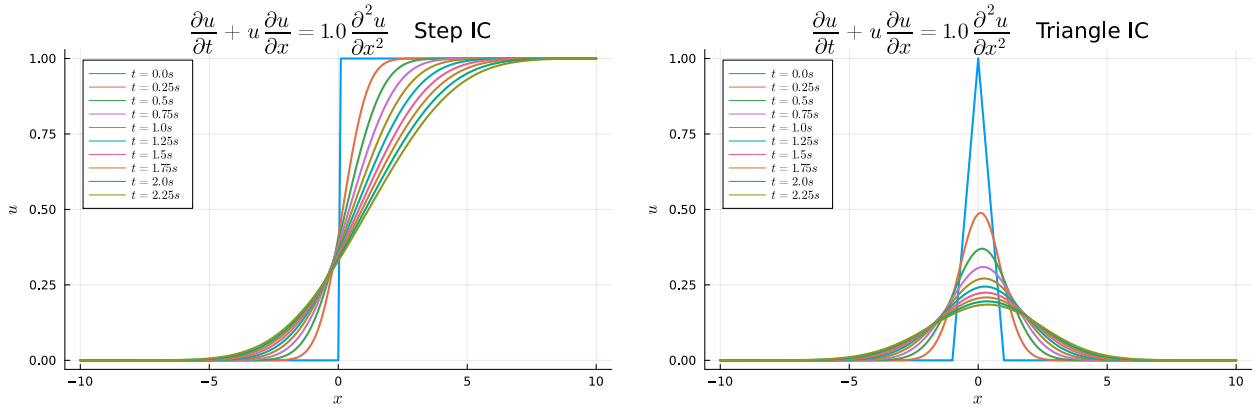
6.1 Burgers 方程特性研究

6.1.1 方程的演化特性

在研究方程演化特性时，我们固定采用非守恒形式中心差分、半隐式格式（如式 2.49中给出的格式），并且给定第一类边界条件：

$$u(-10, t) = \text{Const} = 0 \tag{6.1}$$

考察 Burgers 方程的时间演化特性时，我们选取参数 $\nu = 1.0$ 时，各个时刻的波形图。在图 6.1中为两种不同初始条件下的 Burgers 方程不同时刻的波形分布。



(a) Step IC.

(b) Traingle IC.

图 6.1: Burgers 方程的时间演化特性

图 6.1a表示的是在阶跃初始条件下 Burgers 方程的各时刻波形图，图 6.1b表示的是在三角初始条件下 Burgers 方程的各时刻波形图。与图 5.1中 Heat 方程的波形比较，可以发现二者十分相似，这是因为方程中都有 $u_t = \nu u_{xx}$ 扩散项的成分导致的，因此该方程的一个特性就是有扩散性质，波形扰动会同时向 x 的正负半轴两侧传播。

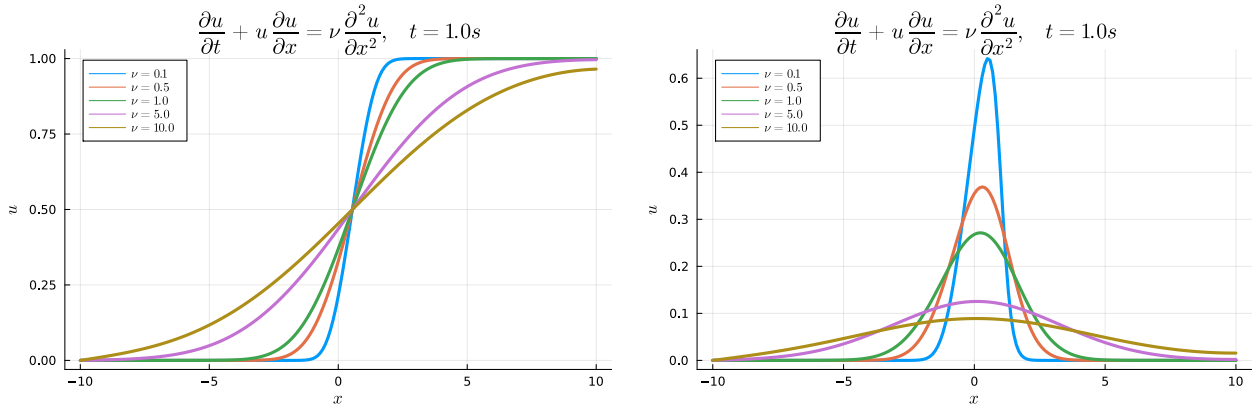
另一方面，方程中有 $u_t + uu_x = 0$ 这对流项一成分，使得整个波形有向 x 正方向偏移，产生不对称性质的趋势。如图 6.1a中，波形中心整体向右侧偏移，图 6.1b中，波形峰值整体向右侧倾斜，波形与图 5.1方程相比多了指向性。

总结 Burgers 方程特性如下：

- Burgers 方程有 Heat 方程的特性，即同时向 x 正负半轴扩散传播；
- 与 Heat 方程不同，在 $u_t + uu_x = 0$ 对流项的存在，扩散过程有向 x 正方向偏向的趋势。

6.1.2 参数 ν 的作用

同前所述，仍然采用固定的非守恒形式中心差分，隐式格式，给定第一类边界条件，选取在 $t = 1.0$ 时刻剖面，参数 ν 选用 $0.1, 0.5, 1.0, 5.0, 10.0$ 五个值，得到波形分布如图 6.2所示。

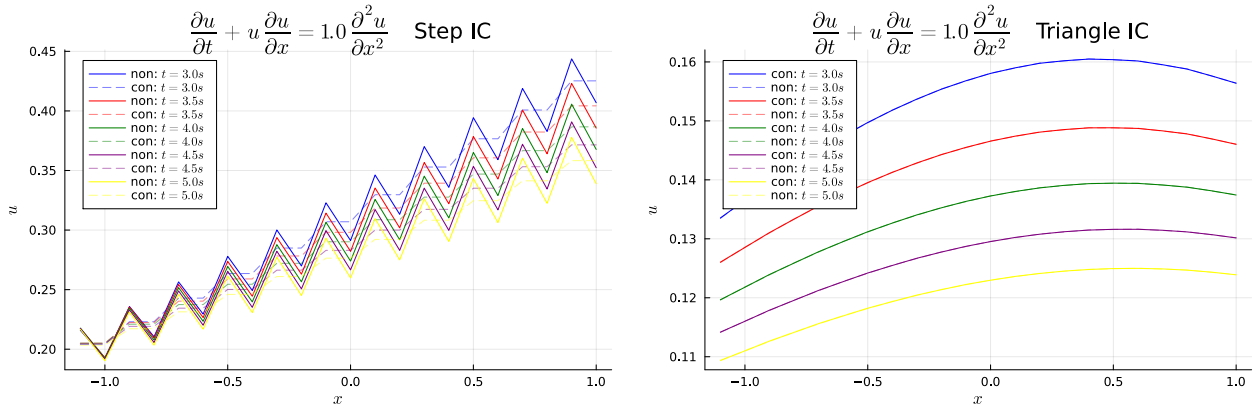


(a) Step IC. (b) Traingle IC.

图 6.2: Burgers 方程参数 ν 的影响

事实上该波形图与 Heat 方程的波形图 5.2也十分类似，得到的结论也类似。图 6.2a表示的是阶跃型初始条件下不同 ν 取值的方程波形，图 6.2b表示的是三角型初始条件下不同 ν 取值的方程波形。由此可见，在差分格式一定时，方程传播固定时长下，参数 ν 取值越大，波形传播扩散得越快，波形被“抹平”的趋势越明显。

6.2 守恒形式与非守恒形式的比较



(a) Step IC. (b) Traingle IC.

图 6.3: Burgers 方程守恒形式与非守恒形式数值解比较

因为在 ν 取 5.0 时，νu_{xx} 这项不满足 CFL 条件，显式数值格式会发散。因此在对比守恒形式与非守恒形式统一选取中心差分格式，显式时间格式，$\nu = 1.0$，考虑在 $t = 3.0 \sim 5.0s$ 时刻守恒形式与非守恒形式的值。

在绘图中发现二者图形几乎一致，无法分出区别，因此绘制图形时截取局部局部区间 $[-1, 1]$ 查看波形，如图 6.3所示。

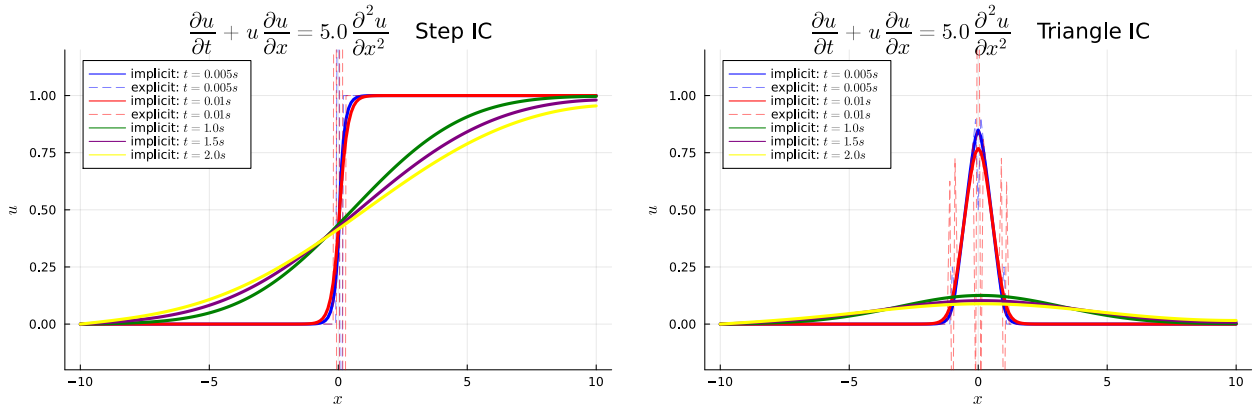
图 6.3a中虚线表示守恒形式数值解，实线表示非守恒形式数值解，显然这个解是存在振荡的（因为采用了显式格式且 CFL 条件正好擦边满足）。但是相对而言，虚线（守恒形式）比实线（非守恒形式）的振荡幅度要小，相对而言更稳定。而图 6.3b在这一段时刻，两个形式的解几乎没有差别。

虽然算例较少，不具有绝对的统计规律。但在此次作业的实践过程中，可视化过程的启示有两点：一方面显式格式下守恒形式与非守恒形式数值解波形差别不大；另一方面守恒形式的数值振荡会比非守恒形式稍微小一些。

6.3 显式格式与半隐式格式的比较

在选取非守恒形式时，显示计算格式（式 2.46）和半隐式计算格式（式 2.49）的数值解波形也有不同。实践表明半隐式解计算格式总是稳定，而显式格式则在 ν 较大时会发散。

为了体现显式格式与半隐式格式的差别，本节考察在第一类边界条件 $\nu = 5.0$ 时的显式与隐式波形。因为在可视化过程中，发现此时显式格式发散得特别快，因此显式格式只取前两步可视化示意，隐式格式取较长演化时间展示，如图 6.4。



(a) Step IC. (b) Traingle IC.

图 6.4: Burgers 方程显式与半隐式格式比较

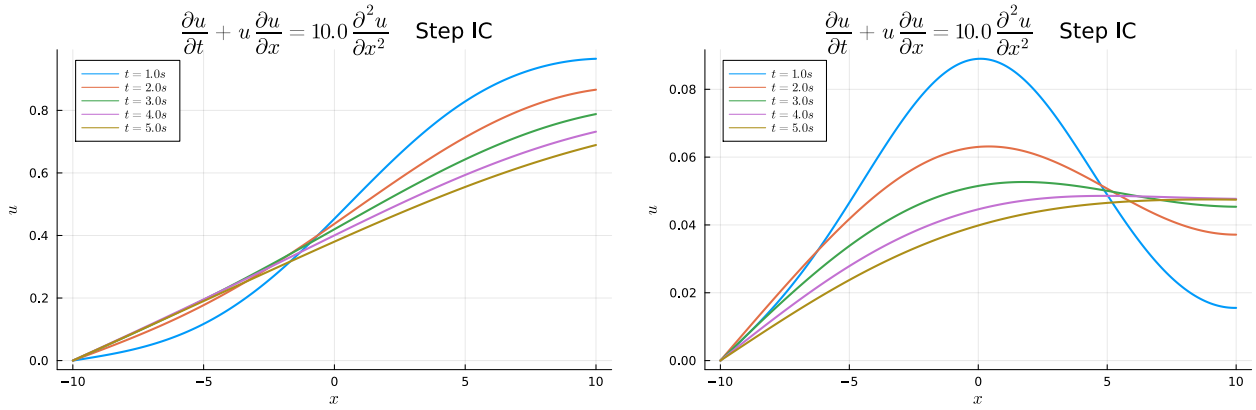
图 6.4a表示的为阶跃初始条件下显式与半隐式格式波形，图 6.4b表示的为三角初始条件下显式与半隐式格式波形。可以发现在 $\nu = 5.0$ 时，显式格式已经发散了，而半隐式格式则可以保持稳定。

上述讨论中表明，Burgers 方程显式格式对 CFL 条件要求更苛刻，显式格式容易发散，无法保证计算收敛。而半隐式格式则能保证数值解的稳定性。

6.4 第一类边界条件的处理与结果

与 Heat 方程类似，第一类边界条件在 $x = -10$ 处给定一个固定的函数，如式 6.1中所述。因为式 2.49中半隐式解可以套用 Heat 方程第一类边界条件处理的方式,，如式 5.6中所述，对计算矩阵进行拆分即可处理边界条件。

为了展现第一类边界条件的施加效果，需要选取 $\nu = 10.0$ 较大情形下的计算结果，这里以空间中心差分格式、时间半隐式格式的计算结果波形作展示。



(a) Step IC. (b) Traingle IC.

图 6.5: Burgers 方程的第一类边界条件计算结果

图 6.5a是阶跃初始条件下的第一类边界条件计算结果，图 6.5b是三角初始条件下的第一类边界条件计算结果。同样的计算结果波形与图 5.4很相似，但 Burgers 方程的解整体有向 x 轴正方向偏移的趋向，这部分是由 $u_t + uu_x = 0$ 对流项所带来的。而第一类边界条件仍然在最左端起到“牵住风筝”的作用。

6.5 小结

经过本章节对于一维 Burgers 方程的有限差分解的讨论，可以得到如下经验和结论：

1. 一维 Burgers 方程的效果是向 x 轴两侧扩散传播，且会有向 x 正半轴倾向的趋势，这部分是由对流项带来的；参数 ν 与 Heat 方程中 α 功效一样，决定了扩散速度。
2. 守恒形式相对非守恒形式的数值振荡幅度更小一些。
3. 半隐式格式的收敛性比显式格式要好许多，至少在本次作业的算例里半隐式格式都是稳定的。

7 本次作业总结与心得

7.1 本次作业工作总结

本次作业针对三个方程形式进行了有限差分解的讨论，针对不同的参数取值、空间差分格式、时间离散格式、边界条件等进行了程序编写与结果比较。

本次作业的程序在考虑了通用性与拓展性的情况下，编写了一套方便更改与调用的一维求解程序。并针对不同情形准备了丰富的算例，源码存放在 example 文件夹下，数据存在 data 文件夹下，gif 图片存在 image 文件夹下。

本次作业报告的撰写也尽量考虑了诸多情形下的算例比较，希望能初步了解有限差分格式这个数值方法的特点，并初步理解计算流体力学的思想。但受制于时间限制，还是会有许多遗漏的地方。另外，在处理第二类边界条件与隐式格式的过程中，我个人数学能力薄弱，可能会有错误。虽然数值计算结果似乎没问题，但希望老师有时间的话能帮忙看一下我的推导是否存在什么谬误。

7.2 本次作业心得

之前觉得一维问题的有限差分解是显然简单的，但在本次作业中还是发现了很多之前没有注意到的问题。比如数值解的稳定性、相容性等，是在没学过具体的理论前很难自己推导出来的。

另外在编程方面，为了能编写足够抽象、复用性好、方便拓展的程序，还是非常复杂的。本次作业我对代码重构了很多次，也在重构过程中对方程有限差分的数值解共性有了更深的认识。

参考文献

[1] Dale A Anderson, John C Tannehill, Richard H Pletcher, Munipalli Ramakanth, and Vijaya Shankar. *Computational fluid mechanics and heat transfer*. CRC Press, Fourth edition. | Boca Raton, FL : CRC Press, 2020. | Series: Computational and physical processes in mechanics and thermal sciences, 2020.

[2] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[3] 张弘博. 求解 burgers 方程的数值方法及其稳定性分析. 2010.