浙江大学

Project

课程名称:	数值计算方法
姓　　名:	胡天扬
学　　号:	3190105708
专　　业:	自动化（控制）
指导教师:	赵豫红

2021　年　4　月　29　日

一、 题目

在制作半导体器件时，常需要在硅晶体中参入杂质来改变其物理特性。在实际生产中，常使用扩散法实现。扩散是物质从高浓度的地方向低浓度的地方转移的现象，将含有杂质的气体包围着硅片，或者将杂质涂抹在硅表面，杂质就会由表及里地扩散。扩散现象可用扩散方程来描述：

$$q = -D\nabla u.$$

其中 q 表示扩散流强度，即单位时间内通过单位横截面积的粒子数；D 表示扩散系数，对于温度恒定的环境，可认为 D 为常数；u 表示粒子浓度（体密度），是关于时间 t 和空间坐标 (x, y, z) 的函数；负号表示扩散的方向与浓度梯度的方向相反。

由于杂质的扩散速度很慢，在研究杂质从硅片的一面向内扩散时，可以不考虑另一面的存在，因此可以将硅片内部建模为半无界空间，即$\{x > 0\}$。从初始时刻 $t = 0$ 开始，杂质从 $x = 0$ 处沿 x 轴方向扩散，考虑如下两种情况：

1、恒定表面浓度扩散。硅片暴露在含有杂质的气体中，杂质的体密度为 ρ，由于气体中杂质充足，在扩散过程中可近似认为 ρ 保持不变。

2、限定源扩散。硅片处于纯净的气体中，在 $t = 0$ 时刻，$x = 0$ 的表面处覆盖着一层杂质，杂质的面密度为 σ。

请针对上述两种情况，分别完成以下问题：

(1) 给出该物理问题的数学描述，即半无界空间下定解问题的数学表达式。

(2) 对于(1)中的问题，给出 u(x, t) 的数值解。其中所有参数（D, ρ, σ）均取 1，且 $0 < t \le 3, 0 < x \le 3$。

(3) ① 对于恒定表面浓度扩散问题，解析解为： $u(x,t) = \rho \mathrm{erfc}\left(\frac{x}{2\sqrt{Dt}}\right)$，

其中 erfc()为余误差函数。

② 对于限定源扩散问题，解析解为： $u(x,t) = \left(\frac{\sigma}{\sqrt{D\pi t}}\right) e^{-\frac{x^2}{4Dt}}$

③ 将解析解与(2)中的数值解对比并分析。

二、 研究背景

物理问题通常研究一个变量 u 相对于时间 t 和空间 r 的变化规律，并且这种变化规律通常以偏微分方程的形式表现出来。（当存在多个变量 u 时，则表现为偏微分方程组）。

一般情况下，这个偏微分方程是二阶的，并且通常可以划分为三个范畴：**波动问题、输运问题、稳定场问题**。这种划分并不绝对，有很多复杂的物理问题不属于这三种范畴，但是这种分类方式足以覆盖大多数常见的物理问题。

1、波动问题研究的是振动的传播规律，其标准表达式为：

$$u_{tt} - \alpha^2 \Delta u = 0$$

其中 u_{tt} 是一种记法，表示 u 相对于 t 的二阶偏导数；a 为一个取决于实际物理参数的常数；$\Delta = \nabla \cdot \nabla$ 是拉普拉斯算子，通常也记为 ∇^2，在直角坐标系下，

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

常见的实例有弦的一维振动，薄膜的二维振动，传输线模型，电磁波等等。

2、输运问题研究的是物质的扩散现象（例如本题）。其标准表达式为：

$$u_t - \alpha^2 \Delta u = 0$$

常见的热传导问题也属于输运问题。

3、稳定场问题研究的是经过足够长时间之后，系统内的稳态分布情况。稳定场问题与时间无关，其标准表达式为：

$$\Delta u = 0$$

常见的实例如稳定的温度、浓度、电流分布等。

从数学上讲，波动方程对应于双曲型偏微分方程；稳定场方程对应于椭圆型偏微分方程；输运方程对应于抛物线型偏微分方程。

在实际应用中，还要注意两点，第一，上述的方程都是齐次的（等式右边为 0），当存在外界干扰因素时（例如受迫振动，存在源点/汇点等），方程通常是非齐次的。第二，还要特别注意“初始条件”和“边界条件”，即系统在 t = 0

时刻所处的状态和系统在边界处所满足的条件，二者分别从时间和空间上限定了系统的原始状态。在偏微分理论中，时间和空间都是普通的变量，不需要刻意区分。因此物理上的初始条件和边界条件在数学上统称为（偏微分方程的）初始条件。

在对偏微分方程的求解上，最常见的解法是**分离变量**。著名的贝塞尔函数就是上述方程在柱坐标系下利用分离变量得到的解，如果是在球坐标系下，则会得到（连带）勒让德方程和球贝塞尔方程。分离变量法通常会得到本征值问题，这对应于贝塞尔函数和勒让德函数中的参数只能取某些特定的值（如整数，半整数等）。分离变量得到的解通常会是无穷级数，而本题中的解析解是闭式解，应利用格林函数得到。

三、 数学模型

由题意，只需考虑水平方向上的扩散：

$$\frac{\partial^2 u}{\partial y^2}=\frac{\partial^2 u}{\partial z^2}=0，\text{即}U=U(x,t)，\text{同时有 } D=\alpha^2。$$

再结合本题的扩散公式和扩散现象的标准表达式：

$$q=-D\nabla u \Rightarrow \frac{\partial u}{\partial x}=-\frac{q}{D} \quad \cdots\cdots ①$$

$$u_t-\alpha^2\Delta u=0 \Rightarrow \frac{\partial u}{\partial t}=D\frac{\partial^2 u}{\partial x^2} \quad \cdots\cdots ②$$

3.1 恒定表面浓度扩散

在①和②的基础上增加边界及约束条件：

$$\begin{cases} u(x,0)=0 & (x>0) \\ u(0,t)=\rho & \\ \lim\limits_{x\to+\infty} u(x,t)=0 & \end{cases}$$

3.2 限定源扩散

由体密度和面密度的关系可知：

$$u = \frac{N}{V} = \frac{S \cdot \sigma}{S \cdot x} = \frac{\partial \sigma(x,t)}{\partial x} \quad \cdots\cdots ③$$

在①、②、③的基础上增加边界及约束条件：

$$\begin{cases} u(x,0) = 0 \qquad (x > 0) \\ \lim\limits_{x \to +\infty} u(x,t) = 0 \\ \int_0^{+\infty} u(x,t)\, dx = \sigma \\ u(0,0) = \left.\frac{\partial \sigma(x,0)}{\partial x}\right|_{x=0} = +\infty \end{cases}$$

四、 计算与结果分析

4.1 算法分析

由于本题有两个未知数 x&t，考虑使用有限差分法构造二维矩阵，利用迭代的思想求解。矩阵维度为 n+1：

$$\begin{pmatrix} u(0,0) & u(0,\Delta t) & \cdots & u(0,n\Delta t) \\ u(\Delta x,0) & \cdots & \cdots & \vdots \\ \vdots & \ddots & \vdots & \vdots \\ u(n\Delta x,0) & \cdots & \cdots & u(n\Delta x,n\Delta t) \end{pmatrix}$$

由此可得递推公式为：

$$u(x_n,t_{n+1}) = u(x_n,t_n) + \frac{\Delta t \cdot D}{(\Delta x)^2}[u(x_{n-1},t_n) - 2u(x_n,t_n) + u(x_{n+1},t_n)]$$

对于恒定表面浓度扩散问题，其初始条件为：

$$\begin{cases} u(0,0) = u(0,\Delta t) = \cdots = u(0,n\Delta t) = \rho \\ u(\Delta x,0) = \cdots = u(n\Delta x,0) = 0 \end{cases}$$

对于限定源扩散问题，其初始条件为：

$$u(0,t_n)=\frac{\sigma}{\Delta x}-\sum_{n=1}^{\infty}u(x_n,t_n)$$

4.2 代码

4.2.1 使用 C 语言求数值解和解析解

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <omp.h>

// 宏定义相关参数
#define ITER_LIMIT (3e5)     //迭代次数上限
#define D 1.0                //D
#define ROU 1.0              //Rou
#define SIGMA 1.0            //Sigma
#define DELTA_X (5e-3)       //x的步长
#define DELTA_T (1e-5)       //t的步长
#define PI 3.1415926536      //pi

// 写入文件的函数
void writeFile(double* data, FILE* file, int times);

int main(void)
{
    // 处理输入
    double xValue, tValue;
    int iterTimes;
    printf("Please input the value of x: ");    scanf("%lf", &xValue);
    printf("Please input the value of t: ");    scanf("%lf", &tValue);
    // 计算迭代次数
    iterTimes = (int)(tValue / DELTA_T) + 1;
    iterTimes = iterTimes > ITER_LIMIT ? ITER_LIMIT:iterTimes;
    // 初始化数组用于存放结果
    double *density0 = (double*)calloc(iterTimes+1, sizeof(double)); //恒定浓度
    double *density1 = (double*)calloc(iterTimes+1, sizeof(double));
    density0[0] = density1[0] = ROU;
    double *source0 = (double*)calloc(iterTimes+1, sizeof(double));  //限定源
    double *source1 = (double*)calloc(iterTimes+1, sizeof(double));
    source0[0] = source1[0] = SIGMA;

    // 进行迭代
    int i, j;
    for( i = 1; i <= iterTimes; i++ )
    {
        // 开启多线程，对限定源问题中除x=0处的点进行加法归约
        double sum = 0.0;
        #pragma omp parallel for reduction(+:sum)
        for( j = 1; j <= i; j++ )
        {
            density1[j] = density0[j] + D*DELTA_T/pow(DELTA_X, 2) * (density0[j-1]-2*density0[j]+density0[j+1]);
            source1[j] = source0[j] + D*DELTA_T/pow(DELTA_X, 2) * (source0[j-1]-2*source0[j]+source0[j+1]);
            sum += source1[j];
        }
        source1[0] = SIGMA - sum;   //计算限定源x=0处的面密度
        // 交换指针进行下一次迭代
        double *temp;
        temp = density0;    density0 = density1;    density1 = temp;
        temp = source0;    source0 = source1;    source1 = temp;
    }
```

```
55      // 将面密度转化为体密度
56      #pragma omp parallel for
57      for(i = 0; i <= iterTimes; i++)
58      {
59          source0[i] /= DELTA_X;
60      }
61
62      // 计算恒定浓度和限定源的理论值
63      double *dstAnalytic = (double*)calloc(iterTimes+1, sizeof(double));
64      double *srcAnalytic = (double*)calloc(iterTimes+1, sizeof(double));
65      iterTimes = (int)(xValue / DELTA_X) + 1;
66      #pragma omp parallel for
67      for(i = 0; i <= iterTimes; i++)
68      {
69          double x = i*DELTA_X;
70          dstAnalytic[i] = ROU * erfc(0.5*x/sqrt(tValue*D));
71          srcAnalytic[i] = SIGMA/sqrt(D*PI*tValue)*exp((-1)*x*x/(4*D*tValue));
72      }
73      //保存数值解和解析解
74      FILE *dstFile, *srcFile;
75      dstFile = fopen("densityData.txt", "w");
76      srcFile = fopen("sourceData.txt", "w");
77      writeFile(density0, dstFile, iterTimes);    writeFile(dstAnalytic, dstFile, iterTimes);
78      writeFile(source0, srcFile, iterTimes);    writeFile(srcAnalytic, srcFile, iterTimes);
79      fclose(dstFile);    fclose(srcFile);
80      printf("Data have been saved. Please use Python script to analyze.");
81
82      return 0;
83  }
84
85  void writeFile(double* data, FILE* file, int times)
86  {
87      int i;
88      for( i = 0; i < times; i++ )
89      {
90          fprintf(file, "%.10f,", data[i]);
91      }
92      fprintf(file, "\n");
93  }
```

考虑到算法的时间复杂度为$O(n^2)$，且由于Δt的取值较小，迭代的次数非常大，耗时很长，因此采用 C 语言进行数值解和解析解的计算。在实际计算中，步长Δx和Δt需自己设定，若步长过小，则运行时间非常长，程序效率低；若步长过大，则会存在不收敛的情况。经过多次手动调整后，最终确定$\Delta x = 5 \times 10^{-3}$，$\Delta t = 1 \times 10^{-5}$。计算完成后分别将恒定浓度问题和限定源问题的结果保存在同目录下的文件中，方便后续的分析。

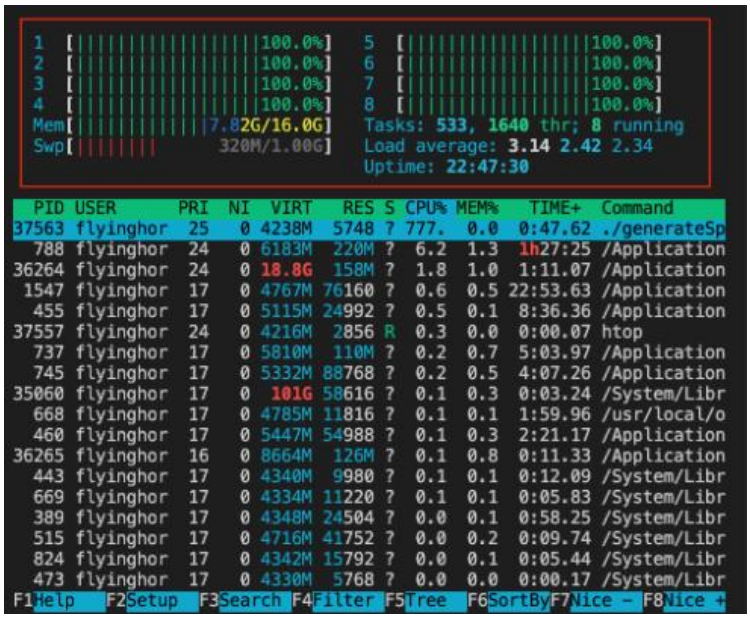
4.2.2 使用 Python 脚本进行数据分析

Python 的 NumPy 库和 matplotlib 库有非常强大的数据处理和可视化能力，在读取 C 语言计算得出的结果后，分别画出两个问题的数值解和解析解曲线，并作出相对误差曲线。

```python
import numpy as np
from matplotlib import pyplot as plt

# 打开文件并读取数值解和解析解
dstFile = open("densityData.txt");  srcFile = open("sourceData.txt")
dstNumerical = dstFile.readline();  dsrAnalytic = dstFile.readline()
srcNumerical = srcFile.readline();  srcAnalytic = srcFile.readline()
dstFile.close();  srcFile.close()

# 数据预处理
dstNumerical = list(map(float, dstNumerical.split(",")))
dsrAnalytic = list(map(float, dsrAnalytic.split(",")))
srcNumerical = list(map(float, srcNumerical.split(",")))
srcAnalytic = list(map(float, srcAnalytic.split(",")))
dstNumerical = np.array(dstNumerical);    dsrAnalytic = np.array(dsrAnalytic)
srcNumerical = np.array(srcNumerical);    srcAnalytic = np.array(srcAnalytic)
x = np.arange(0, 3.005, 0.005)
DPI = 512

# 作图
plt.figure("Constant Density Spread", dpi=DPI)
plt.title("Constant Density Spread")
plt.xlabel("x")
plt.ylabel("u(x,t)")
plt.plot(x, dsrAnalytic, "g-")
plt.plot(x, dstNumerical, "r:", linewidth=2)
plt.savefig("./density.png")

plt.figure("Limited Source Spread", dpi=DPI)
plt.title("Limited Source Spread")
plt.xlabel("x")
plt.ylabel("u(x,t)")
plt.plot(x, srcAnalytic, "g-")
plt.plot(x, srcNumerical, "r:", linewidth=2)
plt.savefig("./source.png")

dstError = np.abs(dsrAnalytic-dstNumerical) / dsrAnalytic * 100
srcError = np.abs(srcAnalytic-srcNumerical) / srcAnalytic * 100

plt.figure("Constant Density Spread Error(%)", dpi=DPI)
plt.title("Constant Density Spread Error(%)")
plt.xlabel("x")
plt.ylabel("Error(%)")
plt.plot(x, dstError)
plt.savefig("./densityError.png")

plt.figure("Limited Source Spread Error(%)", dpi=DPI)
plt.title("Limited Source Spread Error(%)")
plt.xlabel("x")
plt.ylabel("Error(%)")
plt.plot(x, srcError)
plt.savefig("./sourceError.png")

print("Four figures have been saved.")
```

4.3 多核优化

由于数据集非常庞大，在 x 和 t 均取最大值 3 时，需要迭代3×10^5次，约耗时 150s，但是对 C 语言代码开启多线程后仅需 80s，几乎缩短了一半的时间。

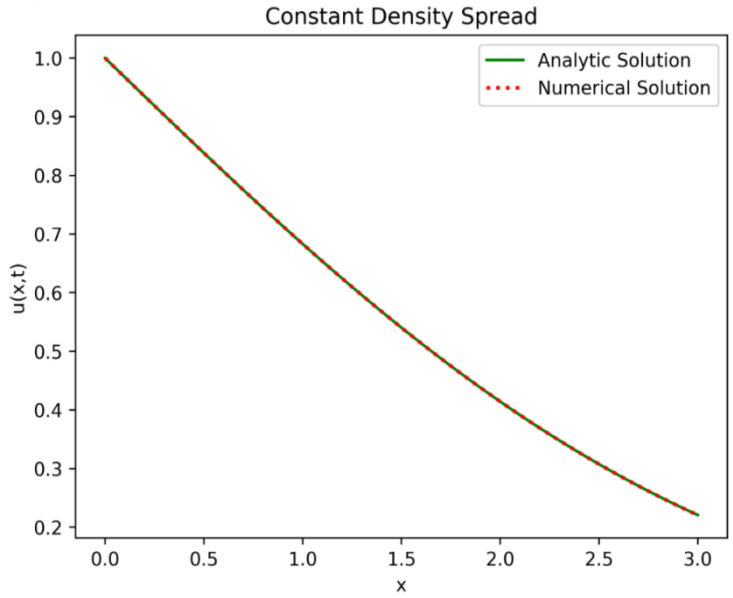


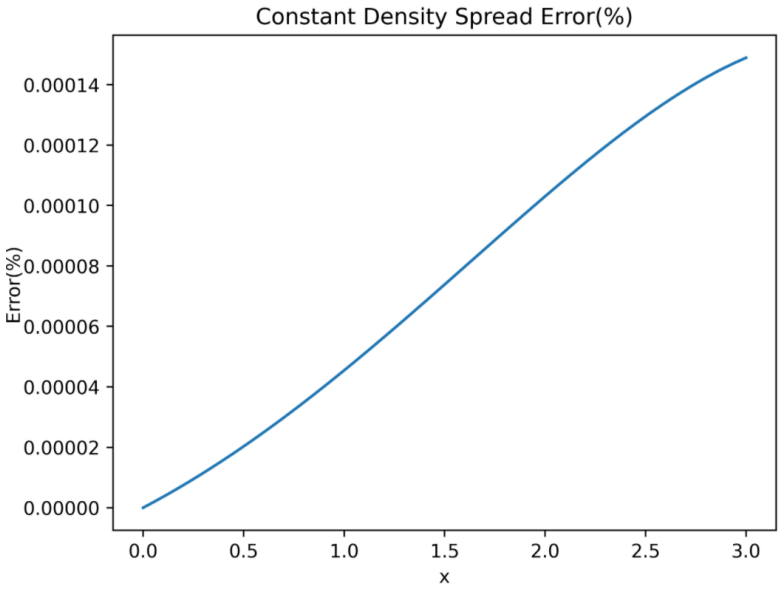
多核优化效果

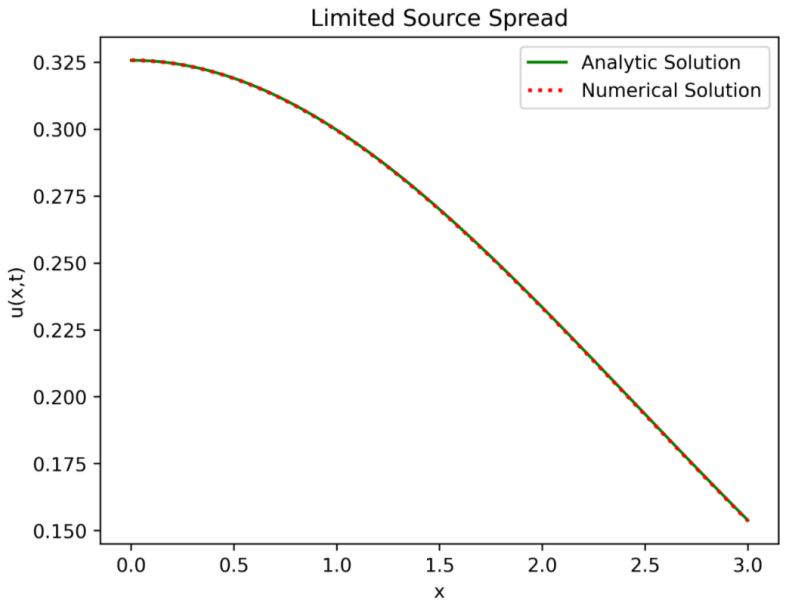
需注意：对于 gcc 编译器，使用命令 ***gcc -O3 -march=native -fopenmp generatData.c -o generateData.exe -lm -std=c99*** 编译；对于 intel 编译器，使用命令 ***icc -O3 -xHost -qopenmp generateData.c -o generateData.exe -std=c99*** 编译。

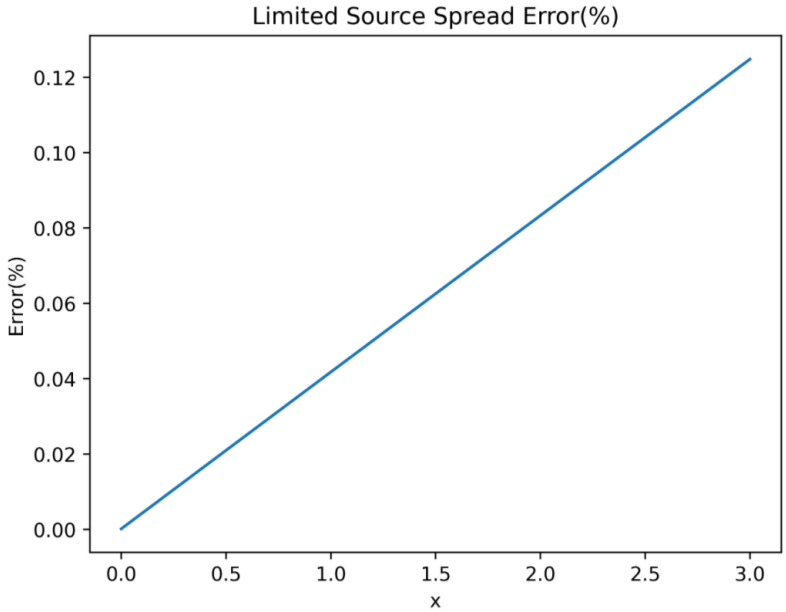
4.4 结果分析

4.4.1 $x = 3, t = 3$




Constant Density Spread Error(%)
Error(%)
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x

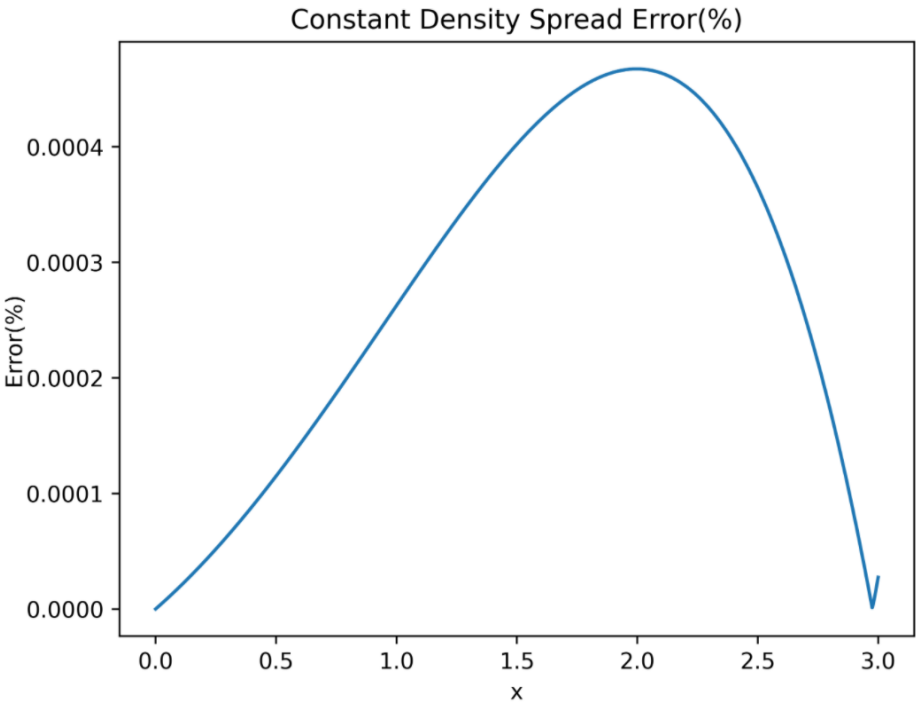

Limited Source Spread
Analytic Solution
Numerical Solution
u(x,t)
0.325
0.300
0.275
0.250
0.225
0.200
0.175
0.150
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x

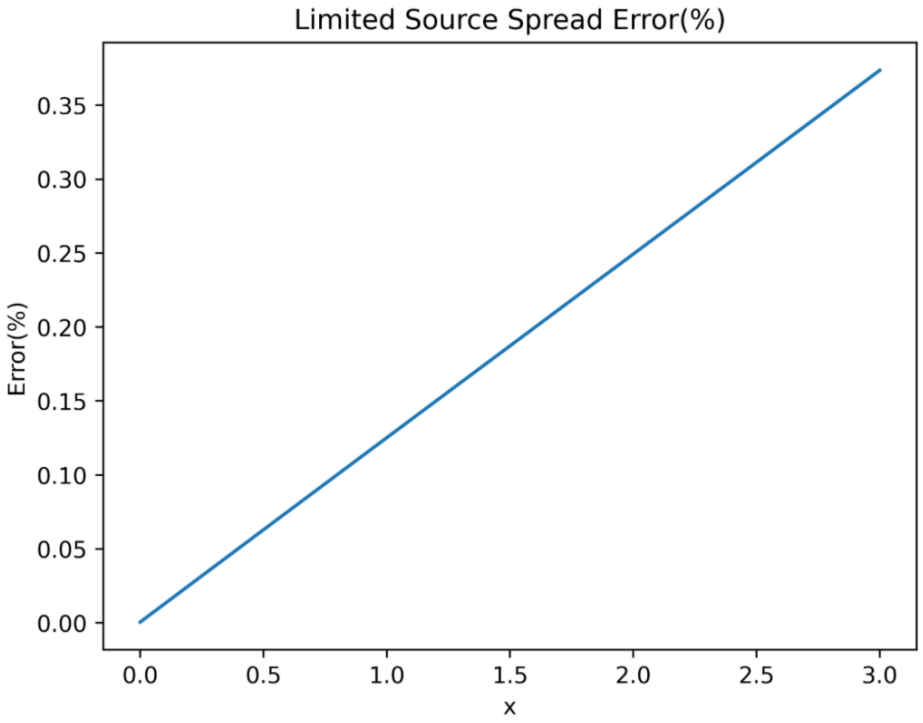

Limited Source Spread Error(%)
Error(%)
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x

图中，绿色实线为理论解析解，红色虚线为数值解，对恒定浓度喝限定源两个问题而言，二者都几乎完全重合，说明数值解的近似模拟效果很好。

蓝线反映了二者的相对误差，随着 x 的增大，两种情况下的相对误差均不断变大，而限定源问题的相对误差似乎呈线性增长的趋势。

4.4.2 $x = 3, t = 1$





改变 t 的值，研究两种情况下的相对误差。恒定浓度扩散时误差并不是持续增大的，而是存在一个最大值，在多次改变参数后发现，在$x = kt$处误差达到最大，k 的值随 t 增大而减小；限定源扩散时误差任随 x 的增大线性增大。这两个现象均是由于其背后的数学公式所决定的，但由于推导较为复杂，不在此赘述。

但恒定浓度扩散对 t 的改变不敏感，相对误差基本被限定在10^{-3}内，而限定源扩散虽然线性程度不变，但是精度会随着迭代次数的增加而显著提升，理论上迭代次数越多相对误差越小。

五、 总结

本题从实际生活中的物理现象出发，将其描述为特定的数学方程，并进行相关问题的求解。

在建立数学模型的过程中，发现题目给出的条件并不足以支撑足够的方程来求解未知量，因此需要根据实际情况自行确定边界条件。

分析算法时利用了有限差分法近似迭代计算的思想，降低了编程的难度，但是由于迭代次数很大，采用了运行速度较快的 C 语言，并进行了多核优化，也可使用 MATLAB 进行运算，它会自动实现多线程计算。

另外在取步长时需要多次试验，比较耗费时间，当 $\frac{\Delta t \cdot D}{(\Delta x)^2}$ 取值合适时，收敛速度会很快，否则就会运行耗时很长或是数列发散。