**PDE数值解法**

抛物线方程（热传导或扩散方程）

椭圆方程（拉普拉斯和泊松方程）

双曲线方程（波动方程）

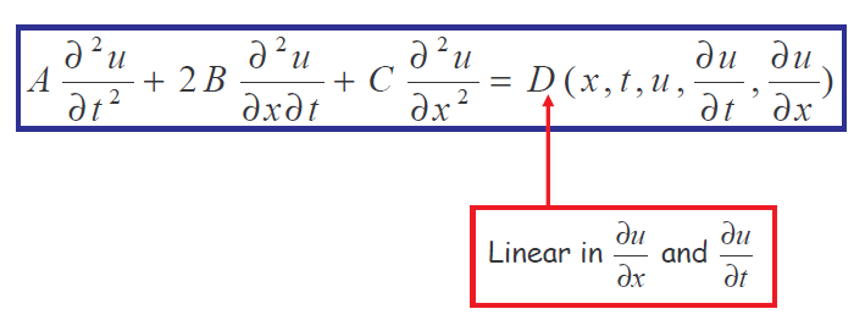
一般来说，涉及多于一个自变量的物理系统，数学模型包含偏微分方程（PDEs）。PDEs广泛应用于流体力学、热传导、弹性、电磁理论、光学、等离子物理、量子力学、金融学等领域。

PDEs通常很难求出解析解，只能退而求其次求其数值解法。有限差分法是求解PDEs的简单且最常用的方法。在该方法中，将PDE中的各种导数用它们的有限差分逼近来代替，则PDE转化为一组线性代数方程，而这些线性方程组可以用一些迭代算法求解。

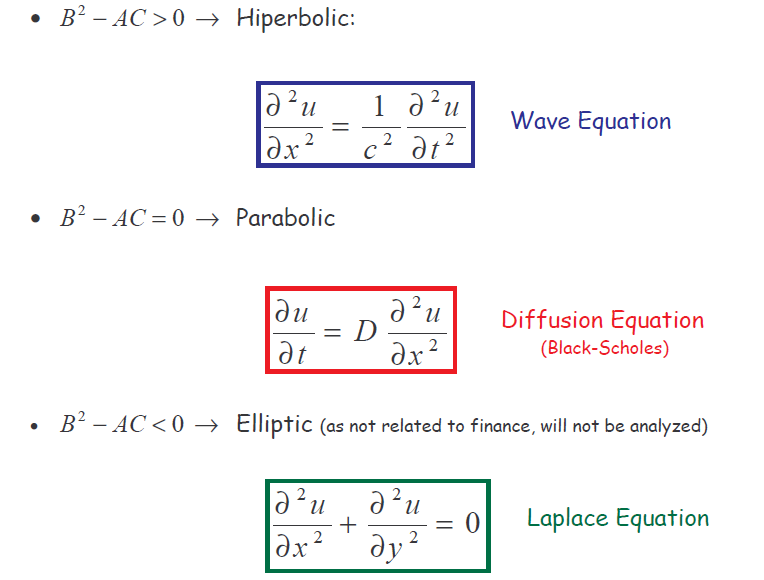
**一．PDE相关理论**

**1. 二阶线性PDEs的分类**

金融领域遇到的PDEs都是线性和二阶的，线性二阶PDE是针对只依赖于两个变量（空间和时间）的二元函数：



可分为三类：

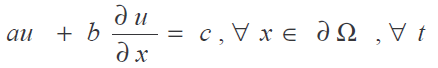


**2. 边界/初始条件**

取决于问题，初始条件为：



以及（或）边界条件：



·若b=0，称为Dirichlet条件；

·若a=0, 称为Newmann条件；

·若c=0, 称为Robin条件。

对于PDE，一般考虑这样几个问题：

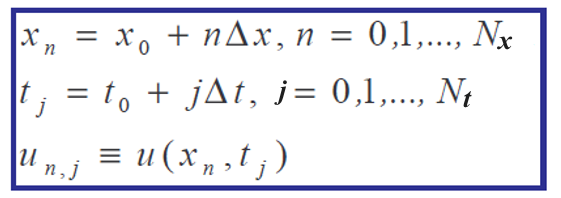
·方程是否有数学意义？应该如何界定边界条件使其为一个适定问题？

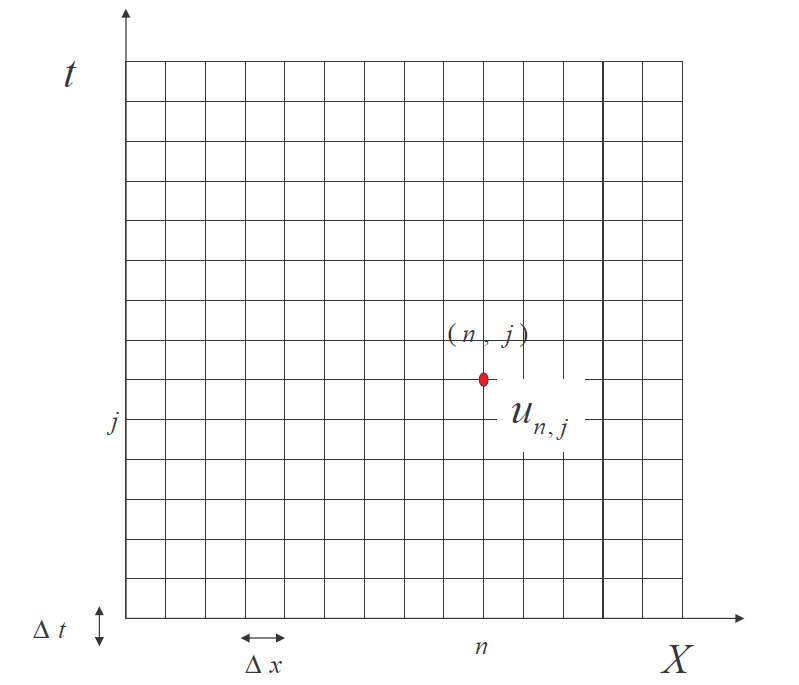
·能否开发出解析工具以求解该问题？

·应该如何求方程的数值解？对于所选的数值方法，解的数学性质有什么涵义？

**3. 二维网格**

**网格**（grid/mesh）定义在(*x*, *t*)空间，满足每个点都可以这样刻画：





**二．平流扩散方程（通量守恒问题）**

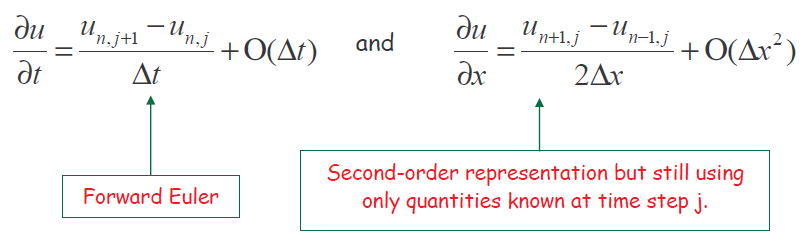
**1. FTCS法（显格式）**

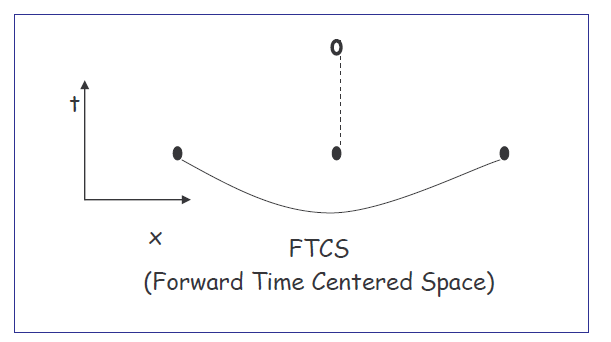
平流扩散方程的一般形式为：



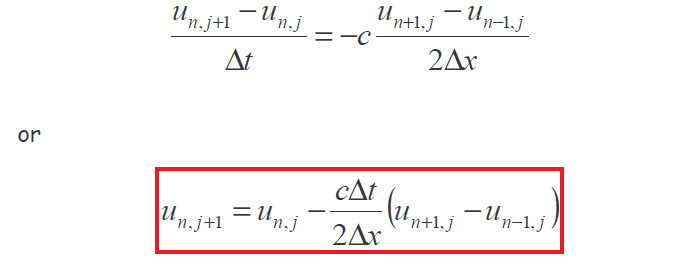
适用于定体积内的粒子守恒：流入有限定量体积的粒子数，等于这些运动粒子流的通量。

利用前向Euler法和只用时间步j处的量的二阶表示：





可得



FTCS算法是一种显格式：对每个n,  可以显式地由已知量计算。后面将介绍隐格式，需要求解关于多个n的隐式方程组得到. FTCS算法也是单水平格式，由于只要知道时间水平j处的值，就能进而计算时间水平j+1处的值。

**2. 代码实例**

% 方程参数及空间,时间范围

Lmax = 1.0; % 最大长度

Tmax = 1.0; % 最大时间

c = 1.0; % 平流速度

% 需要在显式方法内求解的参数

maxt = 3000; % 时间步数

dt = Tmax / maxt;

n = 30; % 空间步数

nint = 15; % 波前: 中间点(u=0)

dx = Lmax / n;

b = c \* dt / (2.0 \* dx);

x = linspace(0, Lmax, n+1);

time = linspace(0, Tmax, maxt+1);

% 初始条件

for i = 1:n+1

if i < nint

u(i,1) = 1.0;

else

u(i,1) = 0;

end

end

% 边界条件

for k = 1:maxt+1

u(1,k) = 1.0;

u(n+1,k) = 0;

end

% 显格式迭代

for k = 1:maxt % 时间步循环

for i = 2:n % 空间步循环

u(i,k+1) = u(i,k) - b \* (u(i+1,k) - u(i-1,k));

end

end

% 绘图

plot(x, u(:,1), '-', x, u(:,10), '-', x, u(:,50), '-', x, u(:, 100), '-')

title('用显式方法I做方波测试')

xlabel('x'), ylabel('Amplitude(x)')

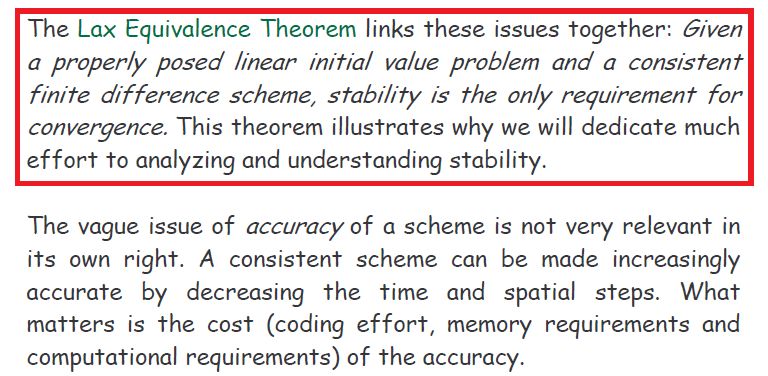


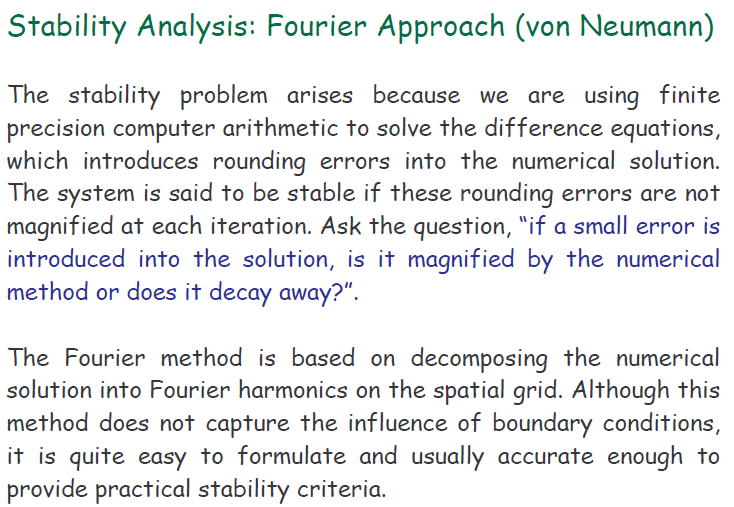
结果很差，是算法无效吗？修改代码中的b值，看结果有多么不稳定！

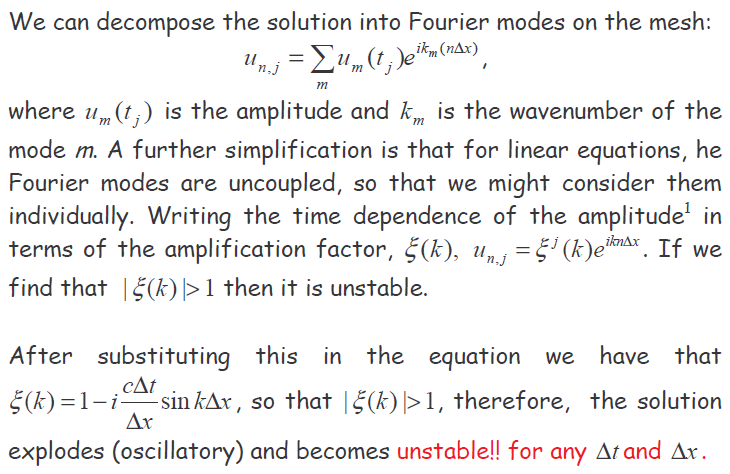
**3. 误差来源与稳定性**

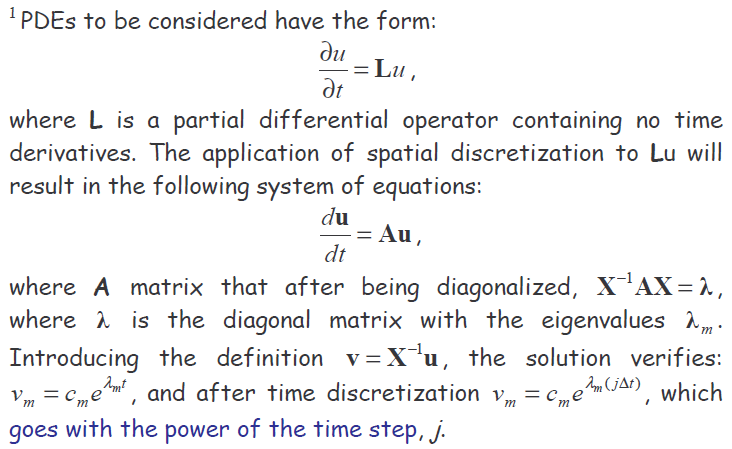
本例中，可以看到有两种基本的误差来源：空间中的截断误差，离散化。截断误差的隐含意思是，数值方式求解的问题与我们想要求解的问题并不完全相同。前面用数值方法得到的PDE近似解可以看作是另一个不同问题的精确解。要刻画数值解做了什么，需要考虑三个基本问题：

1. **一致性**：数值解法称为是一致的，是指当空间和时间步长趋于0时，有限差分格式收敛于PDE；若空间和时间分别独立地做离散化，一致性似乎是没问题的，但当两个离散化混合时，必须要检查一致性。
2. **稳定性**：数值解法称为是稳定的，是指当迭代步数趋于无穷时，数值解与精确解的差是有界的。
3. **收敛性**：数值解法称为收敛的，是指当空间和时间离散化趋于0时（不必相互独立），数值解与精确解的差在指定区域内某固定点是趋于0的。



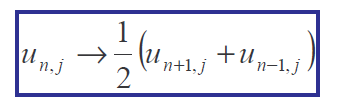




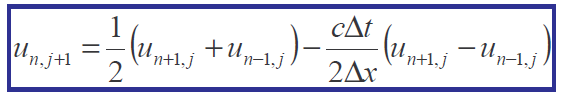


**4. 用Lax方法修正不稳定性**

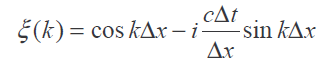
将时间导数中的用其平均做简单代换：



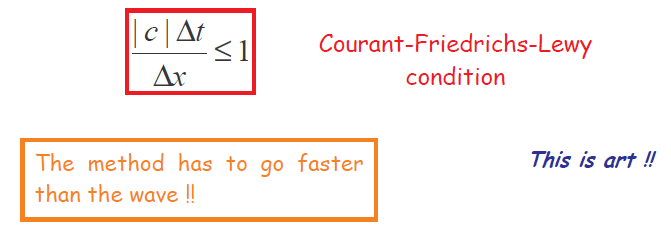
则平流方程变成：



它也是显格式，应用von Neumann稳定性分析，得到：



再由稳定性条件得出稳定性条件为：



Lax法（显格式）求解平流扩散方程代码：

% 方程参数及空间,时间范围

Lmax = 1.0; % 最大长度

Tmax = 1.0; % 最大时间

c = 1.0; % 平流速度

% 需要在显式方法内求解的参数

maxt = 350; % 时间步数

dt = Tmax / maxt;

n = 300; % 空间步数

nint = 50; % 波前: 中间点(u=0)

dx = Lmax / n;

b = c \* dt / (2.0 \* dx); % 若abs(b)<=1/2, Lax方法是稳定的

% b = 0.4268 % 但会扩散, 除非abs(b)=1/2

x = linspace(0, Lmax, n+1);

time = linspace(0, Tmax, maxt+1);

% 初始条件

for i = 1:n+1

if i < nint

u(i,1) = 1.0;

else

u(i,1) = 0;

end

end

% 边界条件

for k = 1:maxt+1

u(1,k) = 1.0;

u(n+1,k) = 0;

end

% Lax方法显格式迭代

for k = 1:maxt % 时间步循环

for i = 2:n % 空间步循环

u(i,k+1) = 0.5 \* (u(i+1,k) + u(i-1,k)) - b \* (u(i+1,k) - u(i-1,k));

end

end

% 绘图

figure(1)

mesh(x, time, u')

title('Lax法中的方波检测')

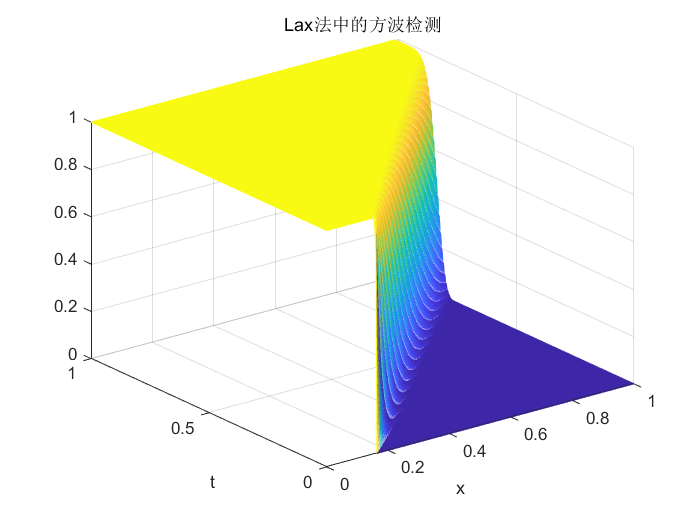
xlabel('x'), ylabel('t')

figure(2)

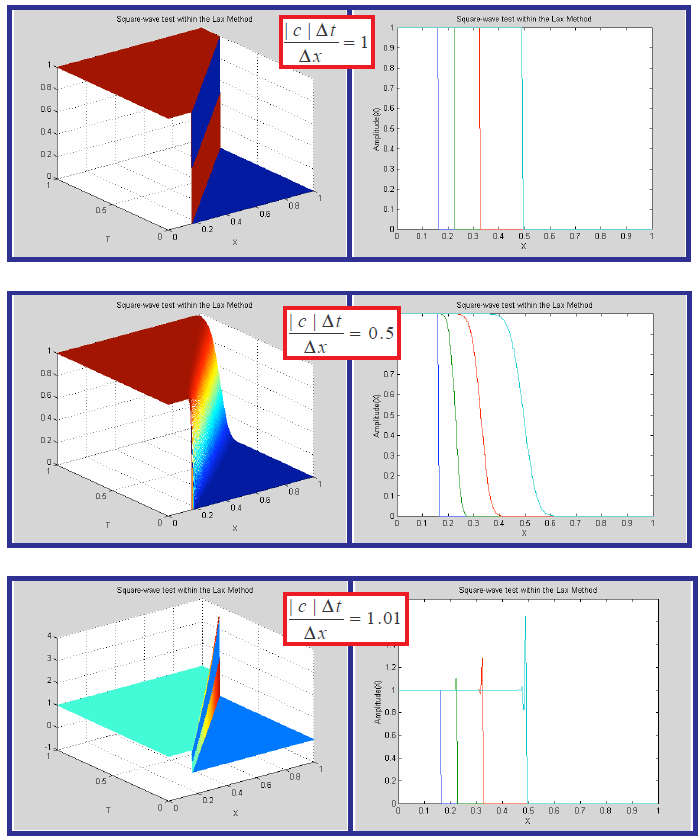
plot(x, u(:,1), '-', x, u(:,20), '-', x, u(:,50), '-', x, u(:, 100), '-')

title('Lax法中的方波检测')

xlabel('x'), ylabel('Amplitude(x)')







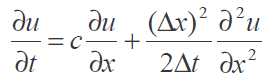
结论：

·若，则显格式法不稳定；

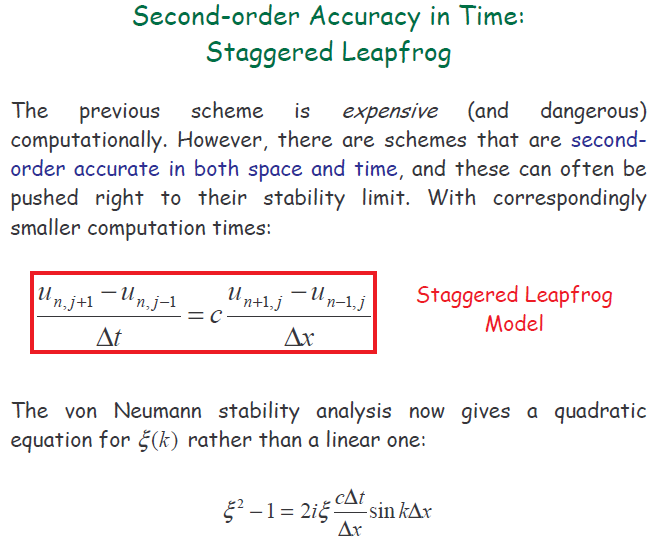
·若，则显格式法扩散（时间步长越小，结果越差）；

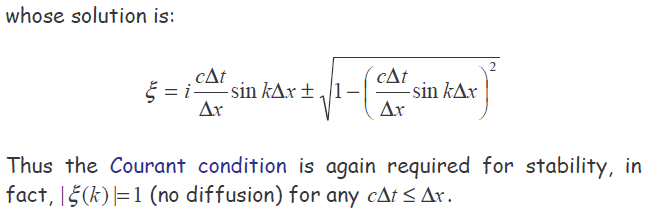
·若，则显格式法收敛到精确解。

**注**：平流方程的Lax法，恰好是如下方程的FTCS法：



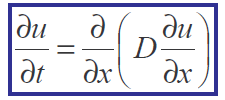
**5. Staggered Leapfrag法**





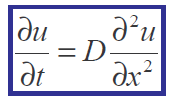
**三．热传导（扩散）方程**

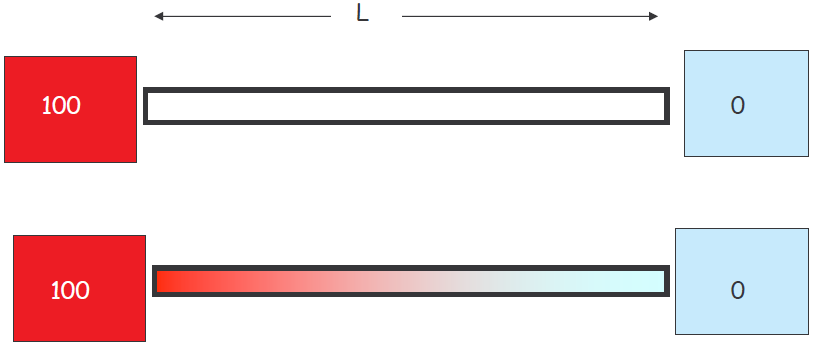
**1. 一维热传导方程**

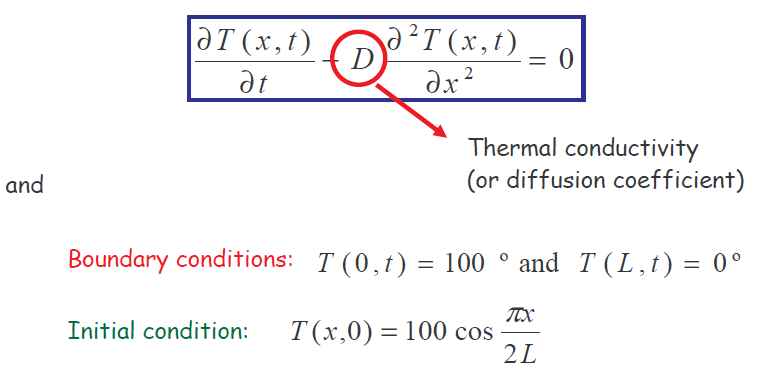


适用于定体积内的热能守恒：定体积内能量（ 温度）的变化，满足Fourier热传导定律，即热流正比于温度的梯度（热导率）。

若D为常数，则有



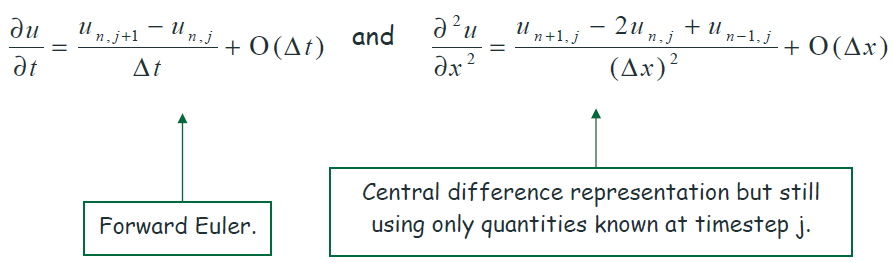




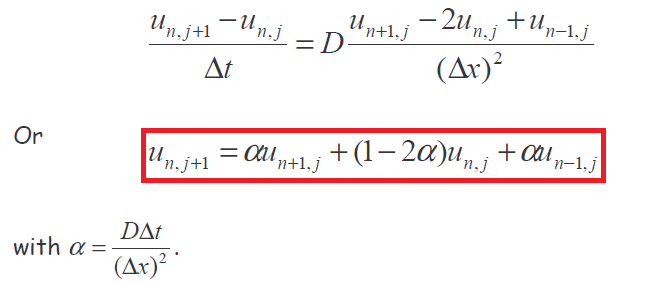
**2. 显式法求解**

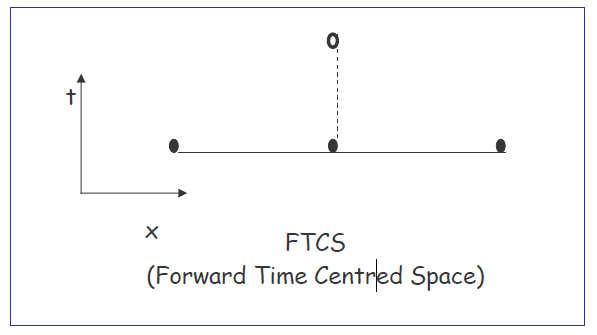
**（1）算法原理**

利用前向Euler法和中心差分表示（只用j时间步已知的量）做迭代：



可得：





**（2）代码实现**

% 热传导方程的参数以及空间和时间范围

L = 1.0; % 总长度

T = 1.0; % 终止时间

% 显式法内中需要求解的参数

maxk = 2500; % 时间步数

dt = T / maxk;

n = 50; % 空间步数

dx = L / n;

cond = 1 / 4; % 热传导率

b = 2.0 \* cond \* dt / (dx \* dx); % 稳定性参数(b<=1)

time = linspace(0, T, maxk+1);

x = linspace(0, L, n+1);

% 初始条件(t=0)

for i = 1:n+1

u(i,1) = sin(pi \* x(i));

end

% 边界条件

for k = maxk+1

u(1,k) = 0;

u(n+1,k) = 0;

end

% 显格式迭代

for k = 1:maxk % 时间步循环

for i = 2:n % 空间步循环

u(i,k+1) = u(i,k) + 0.5 \* b \* (u(i-1,k) + u(i+1,k) - 2.0 \* u(i,k));

end

end

% 绘图

figure(1)

plot(x, u(:,1), '-', x, u(:,100), '-', x, u(:,300), '-', x, u(:,600), '-')

title('显格式迭代求得的温度')

xlabel('x')

ylabel('t')

figure(2)

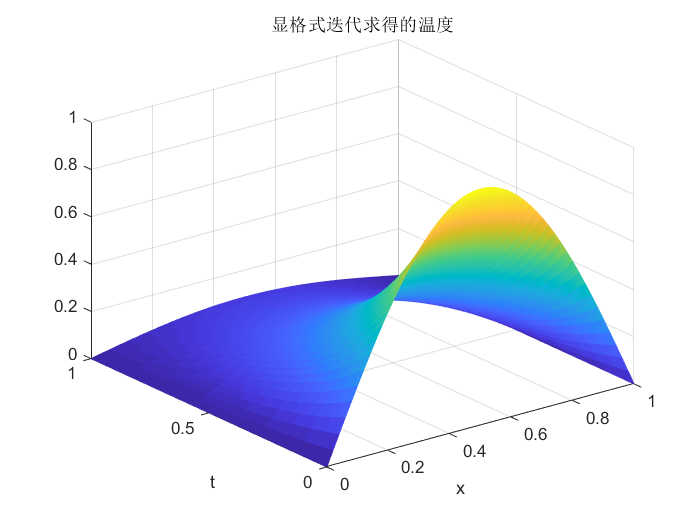
mesh(x, time, u')

title('显格式迭代求得的温度')

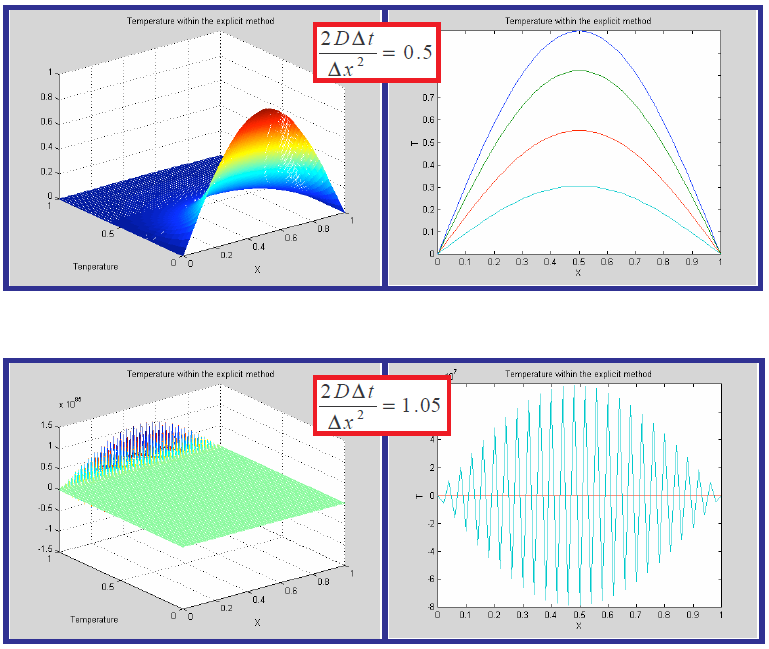
xlabel('x')

ylabel('t')





保持网格不变，修改热传导率：



**（3）稳定性分析（Von Neumann）**

这又是FTCS法，但是多了二阶导。FTCS法对于平流方程（双曲方程）是不稳定的，但尝试形式的独立解，可得到，从而由条件可得稳定性判别准则：

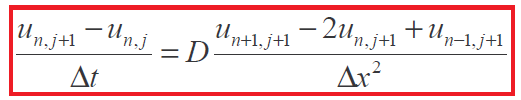


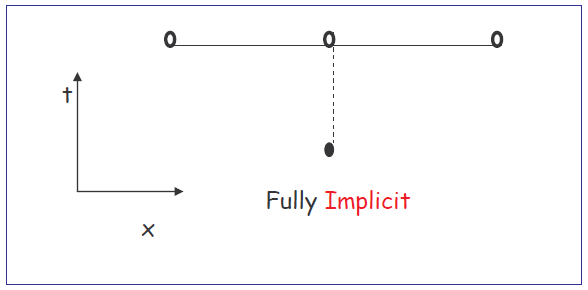
然而，该条件在计算时有很大的局限性。例如，想要分析更细致的空间（），必须要，这就导致要求解就需要非常多的计算步数，从而计算开销巨大。所以有必要考虑新的方法。

**3. 隐式法求解**

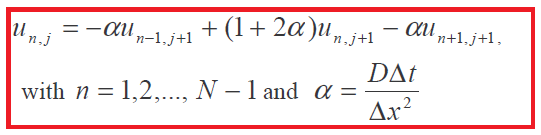
**（1）算法原理**

隐格式迭代仍是FTCS法，只是右边的空间导数是用时间步j+1来计算：



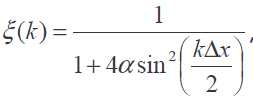


要计算需要在每个时间步求解线性联立方程组，幸运的是这只是个简单问题，因为系数矩阵是三对角线形的：只需要将方程中的项适当分组：



再在n=0和n=N处补充Dirichlet或Neumann边界条件。这些方程随后再深入讨论。

先看稳定性如何？放大因子为



显然对任意的都有. 故隐格式迭代是无条件稳定的。

对于较大的（关于时间只是1阶的）， The details of small-scale evolution form the initial conditions 显然是不准确的，但对于可以得到正确的平衡解。

**（2）代码实现**

% 热传导方程的参数以及空间和时间范围

L = 1.0; % 总长度

T = 1.0; % 终止时间

% 显式法内中需要求解的参数

maxk = 2500; % 时间步数

dt = T / maxk;

n = 50; % 空间步数

dx = L / n;

cond = 1.0 / 4; % 热传导率

b = cond \* dt / (dx \* dx); % 隐式法参数

time = linspace(0, T, maxk+1);

x = linspace(0, L, n+1);

% 初始条件(t=0)

for i = 1:n+1

u(i,1) = sin(pi \* x(i));

end

% 边界条件

for k = maxk+1

u(1,k) = 0;

u(n+1,k) = 0;

end

% 隐格式迭代

aa(1:n-2) = -b;

bb(1:n-1) = 1 + 2.0 \* b;

cc(1:n-2) = -b;

MM = inv(diag(bb,0) + diag(aa,-1) + diag(cc,1));

for k = 2:maxk % 时间步循环

uu = u(2:n,k-1);

u(2:n,k) = MM \* uu;

end

% 绘图

figure(1)

plot(x, u(:,1), '-', x, u(:,100), '-', x, u(:,300), '-', x, u(:,600), '-')

title('隐格式迭代求得的温度')

xlabel('x')

ylabel('t')

figure(2)

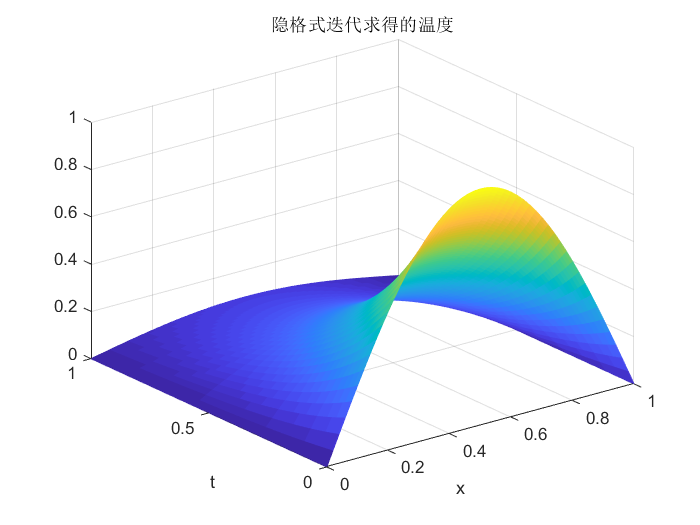
mesh(x, time, u')

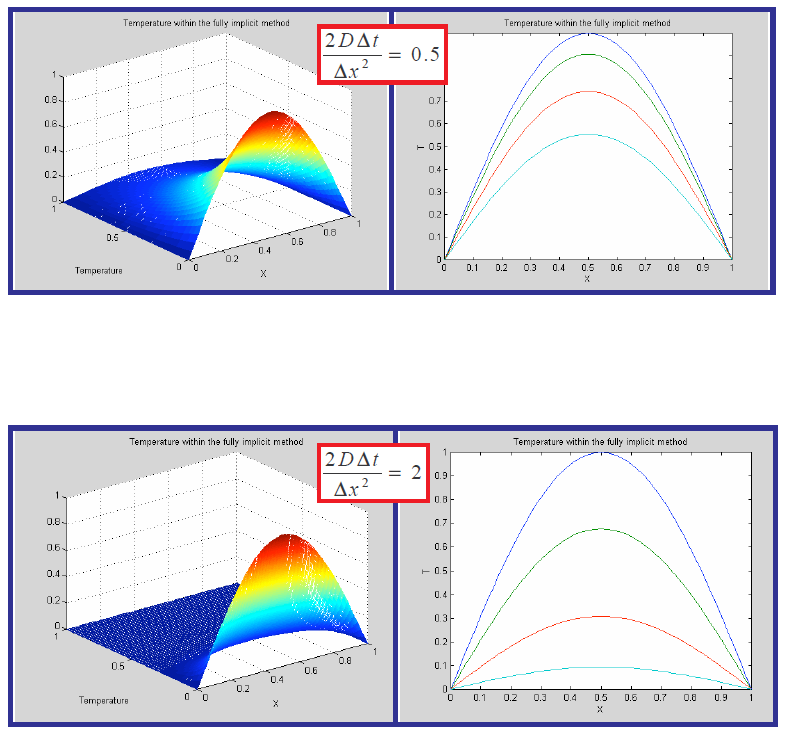
title('隐格式迭代求得的温度')

xlabel('x')

ylabel('t')



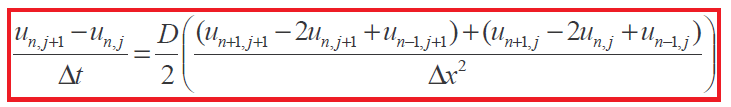


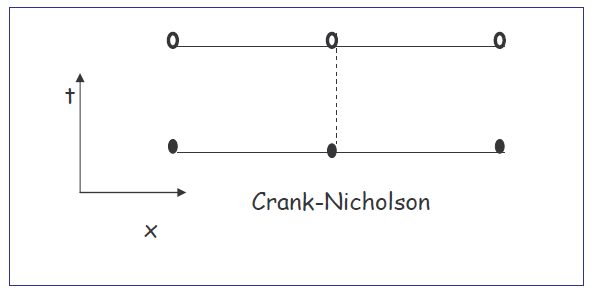


**4. Crank-Nicholson法**

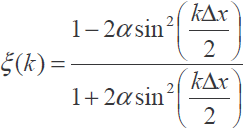
**（1）算法原理**

Crank-Nicholson法是将隐格式的稳定性与空间和时间二阶法的精确性相结合，也可以看成是显式和隐式FTCS法的平均（左边和右边都在时间步j+1/2处做中心化）：





先看稳定性如何？放大因子为



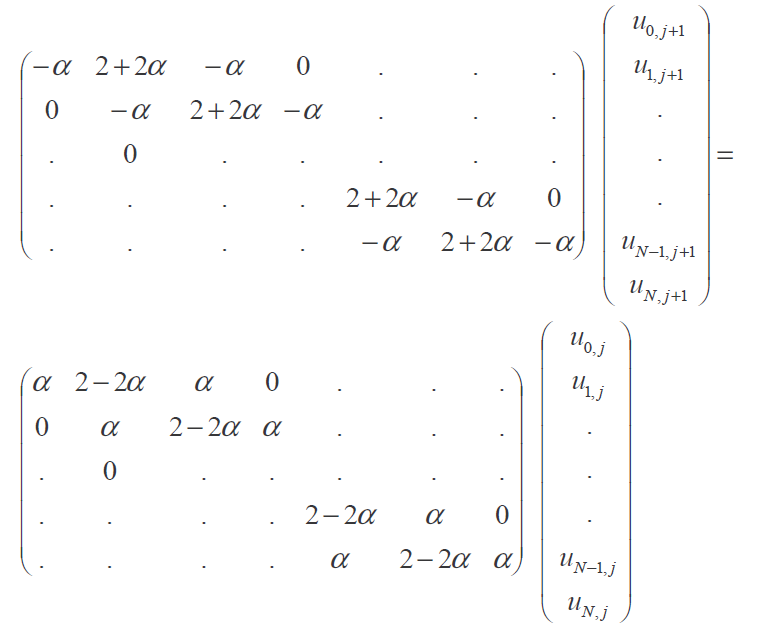
显然对任意的都有. 故Crank-Nicholson法也是无条件稳定的，且关于时间和空间都是二阶的。下面做更深入的分析。

Crank-Nicholson迭代可改写为：



这些方程只对成立。边界条件提供了缺的另两个方程。 这些方程要比显式迭代更难处理，下面讨论它们。

将Crank-Nicholson迭代改写为矩阵形式：

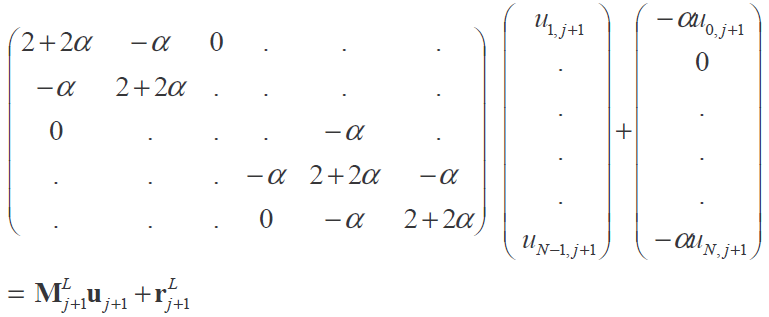


这两个矩阵具有N-1行和N+1列，对应N-1个方程和N+1个未知数，缺的两个方程来自边界条件。利用边界条件，下面将该方程系统转化为方阵表示的方程系统，目标是将方程系统写为如下形式：



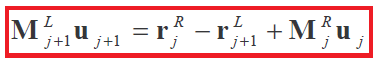
方阵和都已知，向量也已知，其中边界条件的细节已完全包含进来。

**边界条件的例**：若已知和，有时候函数u在边界处（n=0和n=N）的值是已知的，则前文矩阵形式的左边可写为：



右边矩阵也同样处理。

无论哪种边界条件，包含边界条件的Crank-Nicholson迭代格式都可以写成：



如何得到? 从原理上来说，只需对矩阵求逆，则有



然而，矩阵求逆是非常耗时和低效的，后文会介绍两种更好的方式。

**（2）代码实现**

% 热传导方程的参数以及空间和时间范围

L = 1.0; % 总长度

T = 1.0; % 终止时间

% 显式法内中需要求解的参数

maxk = 2500; % 时间步数

dt = T / maxk;

n = 50; % 空间步数

dx = L / n;

cond = 1/2; % 热传导率

b = cond \* dt / (dx \* dx); % CN法参数

time = linspace(0, T, maxk+1);

x = linspace(0, L, n+1);

% 初始条件(t=0)

for i = 1:n+1

u(i,1) = sin(pi \* x(i));

end

% 边界条件

for k = maxk+1

u(1,k) = 0;

u(n+1,k) = 0;

end

% % Crank-Nicholson迭代格式

aal(1:n-2) = -b;

bbl(1:n-1) = 2 + 2.0 \* b;

ccl(1:n-2) = -b;

MMl = diag(bbl,0) + diag(aal,-1) + diag(ccl,1);

aar(1:n-2) = b;

bbr(1:n-1) = 2 - 2.0 \* b;

ccr(1:n-2) = b;

MMr = diag(bbr,0) + diag(aar,-1) + diag(ccr,1);

for k = 2:maxk % 时间步循环

uu = u(2:n,k-1);

u(2:n,k) = inv(MMl) \* MMr \* uu;

end

% 绘图

figure(1)

plot(x, u(:,1), '-', x, u(:,100), '-', x, u(:,300), '-', x, u(:,600), '-')

title('Crank-Nicholson迭代格式求得的温度')

xlabel('x')

ylabel('t')

figure(2)

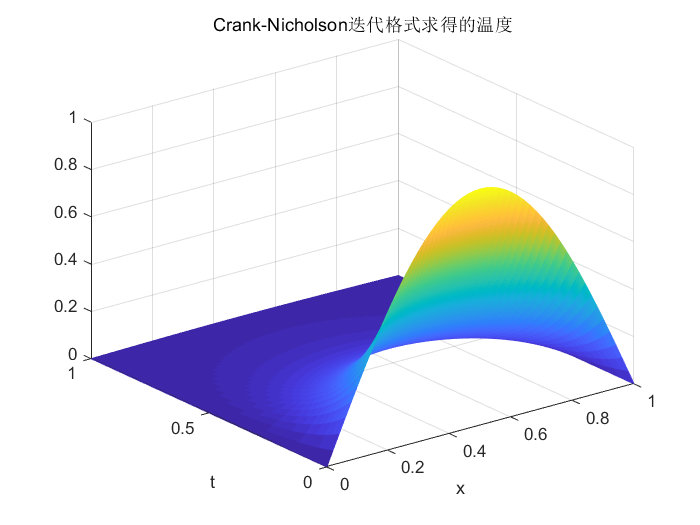
mesh(x, time, u')

title('Crank-Nicholson迭代格式求得的温度')

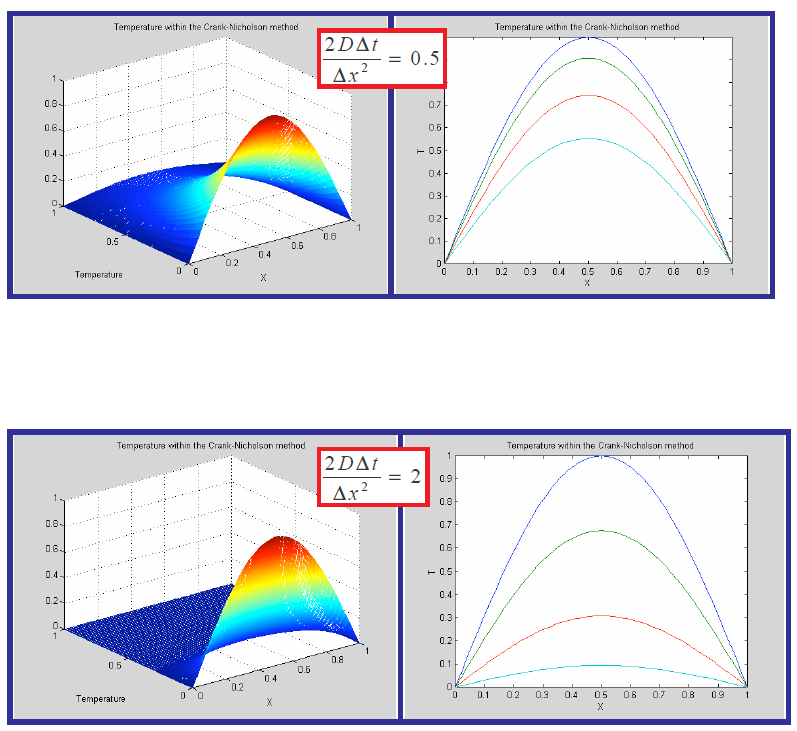
xlabel('x')

ylabel('t')





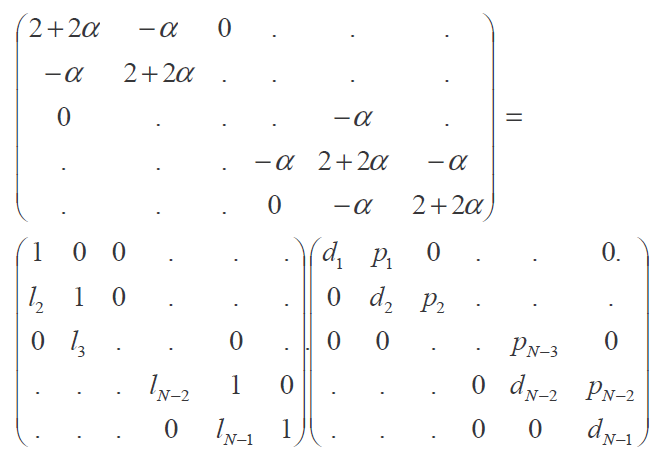
保持网格不变，修改热传导率：



**5. 用LU分解改进Crank-Nicholson法**

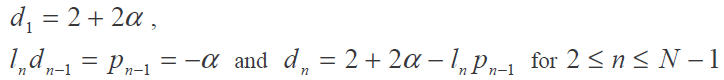
**（1）算法原理**

矩阵是三对角线形，故不难分解为两个矩阵的乘积：下三对角形矩阵（L）和上三对角形矩阵（U），满足M=LU



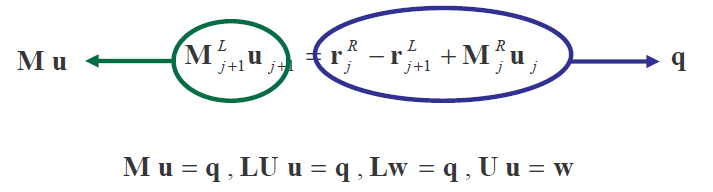
不失一般性，这里选择L的对角元素均为1.

可以验证下面关系成立：



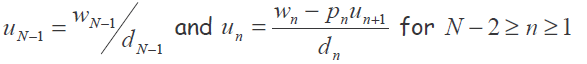
注意是依次从n=1计算到n=N.

现在列出用LU分解法求解的示意图：



接着，很快就要计算完：

* 第1步给出和, 仍是依次计算；
* 第2步涉及到从n=N-2到n=1依次向后计算：



若矩阵方程关于时间独立，则LU分解只需做一次。

**（2）代码实现**

% 热传导方程的参数以及空间和时间范围

L = 1.0; % 总长度

T = 1.0; % 终止时间

% 显式法内中需要求解的参数

maxk = 2500; % 时间步数

dt = T / maxk;

n = 50; % 空间步数

dx = L / n;

cond = 1/4; % 热传导率

b = cond \* dt / (dx \* dx); % CN\_LU法参数

time = linspace(0, T, maxk+1);

x = linspace(0, L, n+1);

% 初始条件(t=0)

for i = 1:n+1

u(i,1) = sin(pi \* x(i));

end

% 边界条件

for k = maxk+1

u(1,k) = 0;

u(n+1,k) = 0;

end

% Crank-Nicholson\_LU迭代格式

aal(1:n-2) = -b;

bbl(1:n-1) = 2 + 2.0 \* b;

ccl(1:n-2) = -b;

MMl = diag(bbl,0) + diag(aal,-1) + diag(ccl,1);

[L,U] = lu(MMl); % LU分解

aar(1:n-2) = b;

bbr(1:n-1) = 2 - 2.0 \* b;

ccr(1:n-2) = b;

MMr = diag(bbr,0) + diag(aar,-1) + diag(ccr,1);

for k = 2:maxk % 时间步循环

uu = u(2:n,k-1);

qq = MMr \* uu;

w(1) = qq(1);

for j=2:n-1

w(j) = qq(j) - L(j,j-1) \* w(j-1);

end

u(n,k) = w(n-1) / U(n-1,n-1);

for i=n-1:-1:2

u(i,k) = (w(i-1) - U(i-1,i) \* u(i+1,k)) / U(i-1,i-1);

end

end

% 绘图

figure(1)

plot(x, u(:,1), '-', x, u(:,100), '-', x, u(:,300), '-', x, u(:,600), '-')

title('Crank-Nicholson迭代格式(LU)求得的温度')

xlabel('x')

ylabel('t')

figure(2)

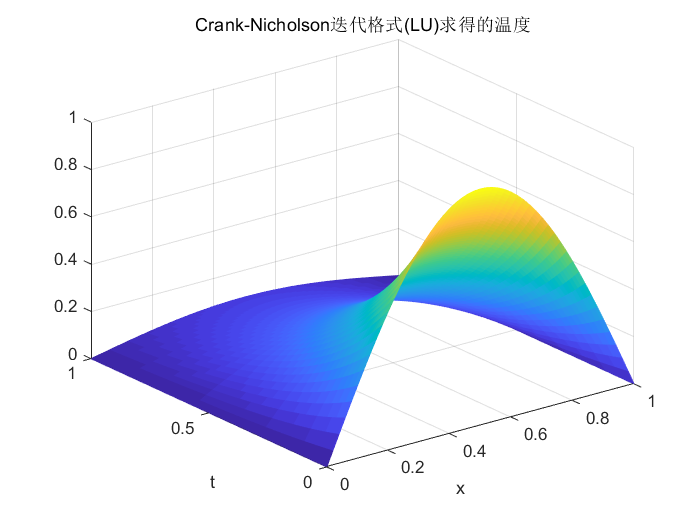
mesh(x, time, u')

title('Crank-Nicholson迭代格式(LU)求得的温度')

xlabel('x')

ylabel('t')





**注**：另一种改进方法是用逐次超松弛法（SOR, Successive Over-relaxation）。