第五次程序作业

PB18010496 杨乐园

2022 年 1 月 3 日

1 问题介绍

对如下一维偏微分方程，基于共轭梯度法与多重网格法对相应的一次有限元空间进行数值求解：

$$\begin{cases} -u^{''} = f \quad 0 \leq x \leq 1 \\ u(0) = 0 \quad u(1) = 0 \end{cases}$$

记录基于两种方法的 CPU 求解时间，并绘制时间与离散程度 N 的点图图像，计算 $MultiGrid$ 方法的时间增长阶，公式如下：

$$growth\ order_{N\to 2N} = \frac{ln(time_{2N})/ln(time_N)}{ln2}.$$

2 实现方法

我们依据代码求解思路顺序给出求解实现程序以及方法的简述：

2.1 有限元变分格式

由等价的弱解形式：

$$Find \quad u_h \in V_h,\ s.t. a(u_n, v) = F(v),\ for\ any\ v \in V_h.$$

其中 $a(u,v) = \int_0^1 u^{'} v^{'}, F(v) = \int_0^1 fv$。当我们假设对 $[0,1]$ 进行 N 等分时，即 $h_e = \frac{1}{N}, x_i = ih_e, i = 0,1,2,..,N$，对于 u_h 在有限元空间内进行系数展开 $u_h = \sum_{i=1}^{N-1} u_i \phi_i(x)$，并依次取有限元空间的基函数 $v = \phi_i, i = 1,2,...,N-1$，从而得到如下方程组：

$$\langle u_n^{'}, \phi_i^{'} \rangle = \sum_{j=1}^{N-1} u_j \langle \phi_j^{'}, \phi_i^{'} \rangle = \langle f, \phi_i \rangle, \quad i = 1,..,N-1$$

将其写成刚度矩阵的形式：

$$K \cdot U = F$$

其中 $K = (\langle \phi_j^{'}, \phi_i^{'} \rangle)_{i,j=N-1}$，$U = (u_1,..,u_{N-1})^T$，$F = (f_1,..,f_{N-1})^T, f_i = \langle f, \phi_i \rangle$。

2.2 有限元空间的基函数以及刚度矩阵的构建

对于一次函数空间，其基函数如下定义：

$$\phi_i = \begin{cases} \frac{x-x_{i-1}}{h_e} & x \in [x_{i-1}, x_i] \\ \frac{x_{i+1}-x}{h_e} & x \in (x_i, x_{i+1}], \quad i = 1, .., N-1 \\ 0 & else \end{cases}$$

基于如上基函数定义，有如下刚度矩阵：

$$K = \frac{1}{h_e} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

2.3 右侧系数向量的构建

对于右侧向量，通过采用复化 3 点 *Gauss* 积分公式计算相关数值积分：

$$\int_a^b f(x)dx = A_1 f(x_1) + A_2 f(x_2) + A_3 f(x_3)$$

其中系数为：

$$A_1 = \frac{5}{18}(b-a) \qquad A_2 = \frac{4}{9}(b-a) \qquad A_3 = \frac{5}{18}(b-a)$$

节点值为：

$$x_1 = \frac{a+b}{2} - \frac{\sqrt{15}}{10}(b-a) \qquad x_2 = \frac{a+b}{2} \qquad x_3 = \frac{a+b}{2} + \frac{\sqrt{15}}{10}(b-a)$$

本此实验将积分区间 16 等分。

2.4 线性方程组解法

2.4.1 共轭梯度法

程序伪代码如下：

```
1 Given matrix A, vector b, initial vector x0, upper limit of the
    number of iterations iteration_max, iteration termination
    precision precision, to solve Ax=b
2 //b.b represents the inner product of b and b
3 up_precision = precision*precision*(b.b)
4 iteration = 0
5 r0 = Ax0
6 r = b-Ax0
7 while (r.r > up_precision && iteration < iteration_times)
8     iteration++;
```

```
        if (iteration == 1)
            p = r;
        else
            beta = r.r / r0.r0
            p = r+beta*p
        alpha = r.r / p.A.p;
        x = x+alpha*p
        r0 = r
        r = r0-alpha*A.p
```

2.4.2 多重网格法

The Full Multigrid Algorithm

```
//multidrid_throw_up represents the projection of the vector to
    the fine grid space
//multidrid_throw_down represents the projection of the vector
    to the coarse grid space
For k=1
    u_1=inverse(A).f_1
For k>=2
    u_k=multidrid_throw_up(u_k-1)
    for i=1:r
        u_k=MG(k,u_k,f_k)
```

The k-th Level Iteration

```
//Given the number of levels k, the initial vector z0, the right
     vector g, MG(k,z0,g) solves A_k.z=g
For k=1
    MG(1,z0,g)=inverse(A_1).g
For k>=2
    //Presmooth step
    form i=1:m1
        z_i=z_(i-1)+(g-inverse(A_k).z_(i-1))/gamma_k
    //Error Correction step
    g_bar=multidrid_throw_down(g-inverse(A_k).z_m1)
    q_0=0
    form i=1:p
        q_i=MG(k-1,q_(i-1),g_bar)
    z_(m1+1)=z_m1+multidrid_throw_ip(q_p)
    //Postsmooth step
```

```
15        from  i=m1+2:m1+m2+1
16              z__i=z_(i-1)+(g-inverse(A_k).z_(i-1))/gamma_k
17        MG(k,z0,g)=z_(m1+m2+1)
```

其中，需要注意的是如下参数选择：

- 矩阵 A_k 为上述刚度矩阵形式，即：设第 k 层为 N 等分区间，则 $A_k = K = N \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$

- 上投影 I_{k-1}^k 的具体形式：设 $k-1$ 层为 N 等分，则 k 层为 $2N$ 等分，并设函数 $v \in V_{k-1}$ 的坐标形式为 $(v_1, v_2, ..., v_{N-1})^T$，记 $w = I_{k-1}^k v$ 则投影生成如下：

$$\begin{aligned} w_1 &= \tfrac{v_1}{2} & w_{2N-1} &= v_{N-1} \\ w_{2i} &= v_i & i &= 1, ..., N-1 \\ w_{2j+1} &= \tfrac{v_j + v_{j+1}}{2} & j &= 1, ..., N-2 \end{aligned}$$

- 下投影 I_k^{k-1} 的具体形式：设 $k-1$ 层为 N 等分，则 k 层为 $2N$ 等分，并设函数 $w \in V_k$ 的坐标形式为 $(w_1, w_2, ..., w_{2N-1})^T$，记 $v = I_k^{k-1} v$ 则投影生成如下：

$$v_i = \frac{w_{2i-1}}{2} + w_{2i} + \frac{w_{2i+1}}{2}, \qquad i = 1, ..., N-1$$

- 参数 Λ_k 的选择。由 $\Lambda_k \geq \rho(A_k)$，并设 k 层区间为 N 等分，则直接取 $\Lambda = 4N$。
- 迭代参数 $r = 100, m1 = 3, m2 = 3, p = 2$。

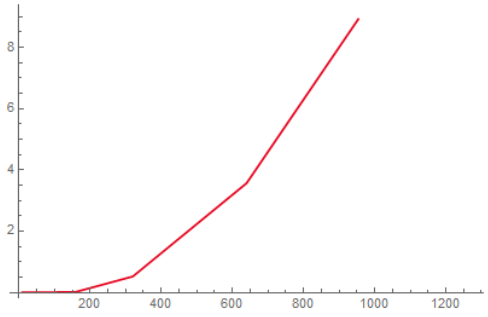
3 程序测试结果

我们选取第一次实验程序的真解函数 $u(x, y) = (x-1)\sin x$ 作为测试程序，测试程序 CPU 运行时间，首先给出 $MultiGrid$ 运行时间以及时间增长阶表格：

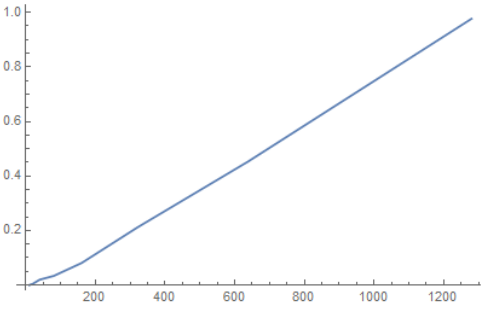
表 1: $MultiGrid$ 运行时间以及时间增长阶表格

n	CPU time	order
10	0.0000758	–
20	0.0082819	6.77162
40	0.0140818	0.765798
80	0.0556167	1.98169
160	0.205971	1.88885
320	0.324359	0.95515
640	0.650401	1.00374
1280	1.25271	0.945651
2560	2.67802	1.09612
5120	5.37822	1.00596

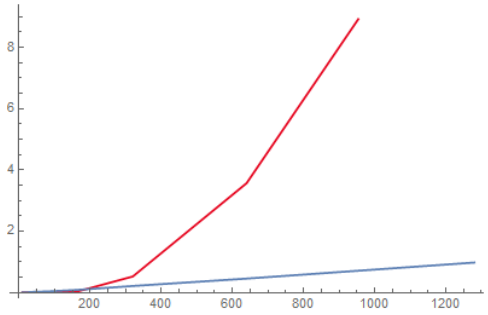
下面给出 CPU 运行时间关于 N 图像如下：



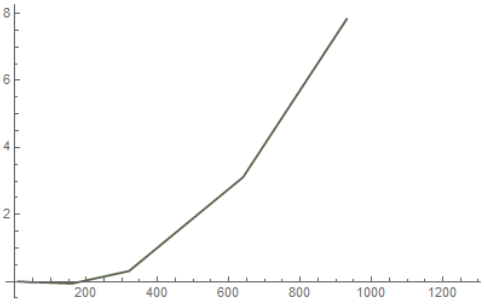
(a) *Conjugate Gradient*



(b) *MultiGrid*



(c) *Together*



(d) *Time gap*

4 结果讨论

通过对数据以及图像的观察我们发现：

随着区间划分 N 的增大，可以明显的看出 *Conjgate Gradient* 方法的求解时间关于 N 成 2 次增长，而 *MultiGrid* 方法求解时间则为线性 1 次，并且明显低于 *Conjugate Gradient* 方法所耗时间。另外，我们可以看到 *MultiGrid* 方法的时间增长阶趋近于 1.0，至于当 N 较小时，可能因为系统开辟空间的原因，所以增长阶不明显，当 N 增大后，开辟空间的时间相对于计算所用的时间较小可忽略不计，所以可以看到时间增长阶满足为 1。当然二者精度阶均为 2 阶，相关计算参见具体的代码 *project* 输出，由于本次重点不在于此，所以不与展示。

5 Computer Code

代码部分请参见附件。