

多重网格法在加速自适应网格 生成中的应用*

刘 卓 曾庆存

(中国科学院大气物理研究所大气科学与地球流体力学国家重点实验室, 北京 100080)

张林波

(中国科学院科学与工程计算国家重点实验室, 北京 100080)

摘 要 多重网格法是 70 年代以来发展起来的用于加速迭代收敛的一种行之有效的办法, 它通过采用不同疏密的网格距来“消除”不同频率范围内的误差分量的办法, 加速了迭代的收敛速度, 具有收敛速度快, 收敛精度高以及节省 CPU 时间等特点。将其用于台风路径预报的自适应网格的生成, 试验表明, 多重网格法比单重网格可提高效率达 20 多倍。

关键词 多重网格法 自适应网格 加速迭代收敛 预报

1 引言

作者在文献[1~4]中介绍了非常定的自适应网格在台风路径预报中的应用。由于网格非常定, 在预报过程中要多次数值生成自适应网格, 因此要求一个代数方程组的解。同样, 如果采用梯形隐式的时间积分及分解算法, 也要求一代数方程组的解。求解代数方程组的一个常用方法是迭代法, 该方法具有方法简单、直观、易于掌握的特点。如 Jacobi 迭代法, Gauss-Seidel 迭代法, 以及 SOR 迭代法等。但对于大型矩阵来说, 迭代法的计算量一般比较大, 且收敛速度比较慢, 甚至不收敛, 这严重地限制了迭代法的应用。在非常定自适应网格模式的数值积分过程中, 迭代是个必不可少的过程。这将使数值积分的计算量大大增加。因此, 如何加快迭代法的收敛速度, 减少计算量, 是衡量非常定自适应网格模式优劣的标准之一。

由于迭代法实际上是对那些相对于网格距是高频的频率范围内的误差分量进行光滑化, 也就是说, 迭代法只能消除高频误差分量, 对低频误差分量的消除能力很弱^[5]。如果在求解过程中, 用一系列逐步加密或减疏的网格去离散求解区域, 而在不同疏密的网格下用迭代法求解, 就可以“消除”不同频率的误差分量, 将所有各重网格上消除误差分量的效果综合起来, 则任何频率范围内的误差分量均可得到消除。这种把离散求解域和用迭代法配合起来的方法, 称为多重网格法 (以后简记为 MG)。

多重网格法是 70 年代以来发展起来的一种比较好的算法, 它具有收敛速度快和计算工作量小的优点, 并且还具有迭代法的简单易于掌握的特点。在偏微分方程的数值解

1996-11-18 收到, 1997-03-25 收到修改稿

* 国家自然科学基金资助项目 49375235

法中,特别是在大型的计算物理和计算流体力学问题的求解中已有大量的应用^[6,7]。

本文介绍了非线性方程的多重网格迭代法,并将其应用于台风路径预报的自适应网格的生成,取得了很好的效果。

2 多重网格法简介

设所要求解的偏微分方程及边界条件为

$$\begin{cases} Lu(x,y) = F(x,y), & (x,y) \in \Omega, \\ \Gamma u(x,y) = \Phi(x,y), & (x,y) \in \partial\Omega, \end{cases} \quad (1)$$

式中 L 、 Γ 是非线性微分算子, u 表示偏微分方程的准确解, Ω 为所求解的区域, $\partial\Omega$ 为区域边界。

用迭代法求解,可以有效地去掉那些与网格尺寸相比是高频的误差分量,以简单的 Poisson 方程为例

$$Lu(x,y) = u_{xx} + u_{yy} = F(x,y). \quad (2)$$

若用 Gauss-Seidel 公式求解,则任一迭代步(例如第 n 步)的数值解 u^n 由下列差分格式求出:

$$\frac{u_{i-1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i+1,j}^n}{\Delta x^2} + \frac{u_{i,j-1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j+1}^n}{\Delta y^2} = F_{i,j}. \quad (3)$$

设第 n 步的未收敛解 u^n 和差分方程的真解 u 之间的误差为 $e^n = u - u^n$, 则其满足齐次的差分方程

$$\frac{e_{i-1,j}^{n+1} - 2e_{i,j}^{n+1} + e_{i+1,j}^n}{\Delta x^2} + \frac{e_{i,j-1}^{n+1} - 2e_{i,j}^{n+1} + e_{i,j+1}^n}{\Delta y^2} = 0. \quad (4)$$

下面我们用 Fourier 分析的方法来研究误差方程(4)的收敛因子。为简单起见,令 $\Delta x = \Delta y = h$, 并假定误差满足周期边界条件, 设某误差分量为

$$e^n = \rho_n e^{i(ph)} e^{i(qh)} = \rho_n e^{i(ph + qh)}, \quad (5)$$

其中 ρ^n 为迭代到第 n 步时误差分量的振幅, p 和 q 分别为 x 和 y 方向上的波数。将其代入误差方程(4), 于是可得收敛因子 γ 为

$$|\gamma| = \left| \frac{\rho^{n+1}}{\rho^n} \right| = \left| \frac{(\cos ph + \cos qh) + i(\sin ph + \sin qh)}{4 - (\cos ph + \cos qh) + i(\sin ph + \sin qh)} \right|. \quad (6)$$

对于波数为 $ph = \pi$, $qh = \pi$ 的高频误差分量来说, 由(6)式可得放大系数为 $|\gamma| = 1/3$, 即每迭代一次, 此类高频误差分量的大小减缩到约为原有的 33%, 若迭代 3 次, 则减缩到仅有约 3.7%, 可见迭代对消除高频误差是很有效的。而对于波数为 $ph = \pi/2$, $qh = \pi/2$ 的中频误差分量来说, 由(6)式可得放大系数 $|\gamma| = \sqrt{1/5}$ 。可见对同一网格在消除中等频率的误差分量时能力相对较弱。对于低频误差分量来说, 例如 $ph \rightarrow 0$, $qh \rightarrow 0$ (即波长 $\rightarrow \infty$), 由(6)式可得 $|\gamma| \rightarrow 1$, 即网格没有能力消除此类频率

的误差分量。因此迭代过程实际上是对高频误差分量的光滑化过程。而低频误差分量要经过许多步数的迭代才能消除。对于波数为 $ph \rightarrow 0, qh \rightarrow 0$ 的低频误差分量将很难消除掉。因此, 对固定的网格距来说, 迭代的大部分步数是为了消除低频误差分量, 从而导致计算量的大大增加, 甚至迭代有可能不收敛。

如前所述, 迭代法只对中高频误差分量起光滑作用, 假设将计算区域划分为 M 重网格, 初始的网格距是 $\Delta x = \Delta y = h_M = h$ 的细网格记为 G^M , 它可消除频率在 $\hat{\rho}\pi \leq (ph, qh) \leq \pi$ 范围内的高频误差分量, $\hat{\rho}$ 为两重网格距之比, 即: $h_k / h_{k-1} = \hat{\rho}$ 或 $h_{k-1} = h_k / \hat{\rho}$, $\hat{\rho}$ 小于 1, 一般 $\hat{\rho}$ 取为 $1/2$ 。设网格距 $\Delta x = \Delta y = h_{M-1} = h / \hat{\rho}$ 的网格为 G^{M-1} 网格, 则它可消除频率在 $\hat{\rho}\pi \leq [p(h/\hat{\rho}), q(h/\hat{\rho})] \leq \pi$ (即 $\hat{\rho}^2\pi \leq (ph, qh) \leq \hat{\rho}\pi$) 范围内的误差分量, 如此继续扩大网格距直到最粗的网格 $G^0: \Delta x = \Delta y = h_0 = h_1 / \hat{\rho} = \dots = h / \hat{\rho}^M$, 则它可消除频率在 $\hat{\rho}\pi \leq [p(h/\hat{\rho}^M), q(h/\hat{\rho}^M)] \leq \pi$ 范围内的误差分量。对于每一重网格 G^k 来说, 频率在 $\hat{\rho}^{k+1}\pi \leq (ph, qh) \leq \hat{\rho}^k\pi$ 范围内的误差分量可以得到消除。将所有网格的光滑效果综合起来, 可得频率在 $\hat{\rho}^{M+1}\pi \leq (ph, qh) \leq \pi$ 范围内的误差分量均得到了消除。由于 $\hat{\rho} < 1$, 则 $\hat{\rho}^{M+1} \ll 1$, 因此多重网格法可在很大的频率范围内不同程度的消除误差。

由于每一重的粗网格均比前一重网格的点数要少 (当 $\hat{\rho}$ 取 $1/2$ 时, 减少到 $1/4$), 因而其计算量也相应地减少, 从而多重网格法的每一个循环过程的工作量和存储量的增加是很小的。

记在细网格 G^M 上求解差分格式为

$$\begin{cases} D^M u^M = F^M, \\ \Gamma^M u^M = \Phi^M, \end{cases} \quad (7)$$

式中 D^M, Γ^M 是网格 G^M 上的离散的差分算子, u^M 是差分方程在 G^M 网格上求解得出的满足收敛标准的收敛解。

求解时, 先给定初始场 u_0^M 。经 s_M 次迭代后, 得到未收敛解 $u_{s_M}^M$, 此时的全场误差即未消除的误差为

$$\varepsilon^M = u^M - u_{s_M}^M.$$

由于迭代只对误差 ε^M 起作用, 而对未收敛解 $u_{s_M}^M$ 并不起作用。如果我们能够求出误差 ε^M 的真解, 则同样可得方程的真解, $u = u_{s_M}^M + \varepsilon^M$ 。因此, 我们的问题转化为求误差 ε^M 的真解。

对于未收敛解 $u_{s_M}^M$ 来说, 误差满足的方程为

$$\begin{cases} D^M(\varepsilon^M + u_{s_M}^M) - D^M u_{s_M}^M = D^M u^M - D^M u_{s_M}^M = F^M - D^M u_{s_M}^M = f^M, \\ \Gamma^M(\varepsilon^M + u_{s_M}^M) - \Gamma^M u_{s_M}^M = \Phi^M - \Gamma^M u_{s_M}^M = \varphi^M, \end{cases} \quad (8)$$

式中 f^M 和 φ^M , 称为 G^M 网格上求解的剩余。由于 D^M 和 Γ^M 是非线性算子, 方程的第一项不可分开。

由于迭代法只能消除相对于网格距是高频的误差分量。当收敛速度变慢时, 说明相对于此网格距来说是高频的误差分量已被消除, 需要过度到粗网格上去消除相对于粗网

格是高频的误差分量 (对细网格而言是中低频)。

记 I_k^{k-1} 为细网格 G^k 到粗网格 G^{k-1} 的限制算子。则将未收敛解 $u_{s_M}^M$ 以及剩余 f^M 和 φ^M 限制到粗网格 G^{M-1} 上去, 可得在 G^{M-1} 网格上满足的方程

$$\begin{cases} D^{M-1}(\varepsilon^{M-1} + I_M^{M-1} u_{s_M}^M) - D^{M-1} I_M^{M-1} u_{s_M}^M = I_M^{M-1} f^M, \\ \Gamma^{M-1}(\varepsilon^{M-1} + I_M^{M-1} u_{s_M}^M) - \Gamma^{M-1} I_M^{M-1} u_{s_M}^M = I_M^{M-1} \varphi^M, \end{cases} \quad (9)$$

其中, 算子 D^{M-1} 和 Γ^{M-1} 是定义在 G^{M-1} 网格上的差分算子, 在形式上与 D^M 和 Γ^M 相同。

对于任意重网格 $G^k (k < M)$, 令

$$u^k = \varepsilon^k + I_{k+1}^k u_{s_{k+1}}^{k+1}, \quad (10a)$$

$$F^k = D^k I_{k+1}^k u_{s_{k+1}}^{k+1} + I_{k+1}^k (F^{k+1} - D^{k+1} u_{s_{k+1}}^{k+1}), \quad (10b)$$

$$\Phi^k = \Gamma^k I_{k+1}^k u_{s_{k+1}}^{k+1} + I_{k+1}^k (\Phi^{k+1} - \Gamma^{k+1} u_{s_{k+1}}^{k+1}). \quad (10c)$$

则在网格 G^k 上满足的方程为

$$\begin{cases} D^k u^k = F^k, \\ \Gamma^k u^k = \Phi^k, \end{cases} \quad (11)$$

在 G^k 网格上求解 (11) 式, 求解时的初始场为 $I_{k+1}^k u_{s_{k+1}}^{k+1}$ 。迭代 s_k 次后, 得出未收敛解 $u_{s_k}^k$, 若求解时收敛变慢, 则再过渡到更粗的网格上去求解。

收敛速度变慢的定义如下, 首先计算出迭代 p 次后全场各网格点上的剩余, 取其中的最大剩余, 将前后两次迭代求解所得的最大剩余之比以 δ 表示, 即

$$\delta_D = \frac{\|D^k u_p^k - F^k\|_\infty}{\|D^k u_{p-1}^k - F^k\|_\infty}, \quad \delta_\Gamma = \frac{\|\Gamma^k u_p^k - \Phi^k\|_\infty}{\|\Gamma^k u_{p-1}^k - \Phi^k\|_\infty},$$

若 δ 大于或等于某个小于 1 的常数 η , 比如 $\eta = 0.6$, 则认为收敛速度变慢。

现在的问题转化为在 G^k 网格上求解 (11) 式, 该式与 (7) 式在形式上完全一样, 只不过网格的疏密程度不一样。迭代 s_k 次得到的未收敛解 $u_{s_k}^k$ 是由两部分组成的, 一部分是 G^{k+1} 网格上的未收敛解在 G^k 网格上的插值, 另一部分是在 G^k 网格上对 $I_{k+1}^k u_{s_{k+1}}^{k+1}$ 的修正。当过渡到最粗的网格上消除了绝大部分误差后, 得出的是收敛解 u^0 , 由此可以求出 G^0 网格上对 $I_{s_1}^0 u_{s_1}^1$ 的修正 ε^0 : $\varepsilon^0 = u^0 - I_{s_1}^0 u_{s_1}^1$ 。

对于任意重网格 $G^k (k > 0)$, 记 I_{k-1}^k 为粗网格 $G^{(k-1)}$ 到细网格 G^k 的插值算子。则可将上一层粗网格 $G^{(k-1)}$ 上的修正量插值到 G^k 网格上作为对原有解 $u_{s_k}^k$ 的修正, 得出 G^k 网格上的收敛解 u^k :

$$u^k = u_{s_k}^k + I_{k-1}^k (u^{k-1} - I_{s_k}^{k-1} u_{s_k}^k). \quad (12)$$

每当由粗网格 G^{k-1} 过渡到细网格 G^k 修正了原有的解以后, 修正后的新解 u^k 若满足方程 (11), 则认为 u^k 就是方程 (11) 的收敛解。由 (12) 式再向更细的网格 G^{k+1} 重过渡, 直到最细网格 G^M 的原有的解 $u_{s_M}^M$ 得到修正后, 求出差分方程在 G^M 网格上的收敛解。

若在修正细网格 G^k 的解后, u^k 不满足方程(11),说明还有未消除的误差。则依照(10)式再插值到粗网格上去消除误差。

由粗网格 G^k 过渡到细网格 G^{k+1} ,是将 G^k 网络上得到的修正量亦即 G^k 网络上消除的误差 e^k 插值到 G^{k+1} 网格上去,而不是直接将全近似 $u^k = e^k + I_{k-1}^k u_{s_{k-1}}^{k+1}$ 插值到细网格 G^{k+1} 上去,这一点是很重要的,因为迭代过程只是对修正量即误差起光滑作用,因此在粗网格 G^k 上得到的对 $I_{k+1}^k u_{s_{k+1}}^{k+1}$ 的修正量 e^k 可插值到细网格 G^{k+1} 上, $I_{k+1}^k e^k$ 作为对 G^{k+1} 网格上的已有的未收敛解 $u_{s_{k+1}}^{k+1}$ 的修正量 e^{k+1} ,这就是MG方法的特点。

MG方法计算流程用框图来说明,如图1。

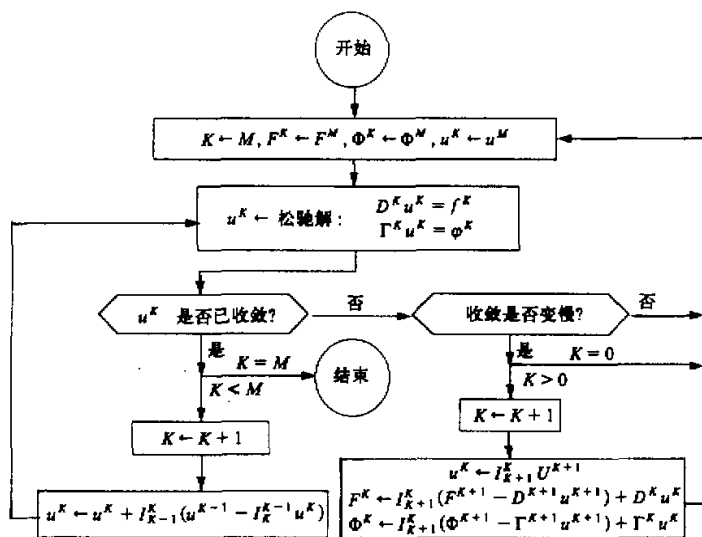


图1 非线性问题的多重网格算法流程图

本文介绍的多重网格法适应于一般的非线性方程,特别是适用于非均匀网格(包括自适应网格)的坐标变换方程(非线性方程)。

3 多重网格法在加速自适应网格生成中的应用

在文献[2]中给出了自适应网格坐标变换方程(见文献[2]中(5)式)

$$\begin{cases} b_1 x_{\xi\xi} + b_2 x_{\xi\eta} + b_3 x_{\eta\eta} + a_1 y_{\xi\xi} + a_2 y_{\xi\eta} + a_3 y_{\eta\eta} + \lambda_w J^2 \frac{\partial W}{\partial x} = 0, \\ a_1 x_{\xi\xi} + a_2 x_{\xi\eta} + a_3 x_{\eta\eta} + c_1 y_{\xi\xi} + c_2 y_{\xi\eta} + c_3 y_{\eta\eta} + \lambda_w J^2 \frac{\partial W}{\partial y} = 0, \end{cases} \quad (13)$$

式中的系数 a_i, b_i, c_i 均为坐标 (x, y) 对 (ξ, η) 的一阶导数的二次项。 λ_w 是常数, J 为坐标变换的Jacobi项, $W = W(x, y)$ 是控制网格疏密程度的权重函数。用算子的形式写出就是

$$\begin{cases} L_1 u = Bx + Ay + \lambda_u J^2 \frac{\partial W}{\partial x} = 0, \\ L_2 u = Ax + Cy + \lambda_v J^2 \frac{\partial W}{\partial y} = 0, \end{cases} \quad (14)$$

式中 $u = (x, y)^T$, A 、 B 、 C 皆为非线性偏微分算子, $W = W(x, y)$ 是 x 、 y 的非线性函数, 因此 L_1 、 L_2 是非线性算子。设 $L = (L_1, L_2)^T$, 则 (14) 式简记为

$$Lu = F = 0. \quad (15a)$$

式中, $F = (F_1, F_2)^T$ 。由于网格点在边界上是已知的, 因此

$$u|_{\partial\Omega} = \varphi(\xi, \eta). \quad (15b)$$

式中, $\varphi = [\varphi_1(\xi, \eta), \varphi_2(\xi, \eta)]^T$ 为一已知函数。

(15) 式构成了求解自适应网格坐标变换的基本方程, 这是一个非线性的椭圆型方程的边值问题, 可以采用多重网格法来加速迭代的收敛。由于 L 是非线性算子, 正好需要用到本文介绍的 MG 方法求解 (15) 式。

在每一重网格上, 迭代采用 Gauss-Seidel 迭代法, 从细网格到粗网格采用的插值是保留格点上的 u 、 F 、 Φ 等与网格点一起保留, 删去的网格点上的 u 、 F 、 Φ 等与网格点一起删去。从粗网格到细网格的插值采用双线性插值, 收敛标准为前后两次算出的 u 之差小于 10^{-4} 。利用这个方案, 本文进行了一项数值实验, 即在一个矩形区域内的某个局部区域网格加密的自适应网格的生成, 该网格可用于台风路径的数值预报, 所生成的网格如图 2, 网格最密处即为台风中心的位置。

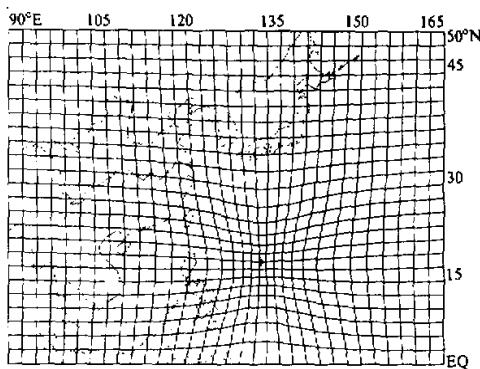


图 2 用于台风路径预报的自适应网格

计算在 Convex C120 上进行。迭代共进行了 6 步循环 (每一步循环定义为从细网格到最粗网格, 再由最粗网格回到细网格), 共花费 CPU 时间 7.64 s。

为了进行比较, 本文还用 SOR 迭代法来数值生成该自适应网格。对于同样的收敛标准及权函数 $W(x, y)$, SOR 法共进行了 717 步迭代才达到收敛, 共花费 CPU 时间 164.47 s。

从效果上来看, 多重网格法所花费的 CPU 时间只是 SOR 法的 1/22, 图

3 给出了两种算法的收敛情况。图中表明多重网格法的收敛速度比 SOR 法要快得多。从图中还可看出, SOR 法在开始时误差下降较快, 随后下降很慢, 这说明高频误差很快被消除, 而其主要工作量都花费在消除低频误差上了。

试验还表明, 多重网格法与单重网格相比有以下几个特点:

- 收敛速度快, 计算量小。每一次迭代循环的工作量只相当于单重网格的 4/3 倍 (对二维问题而言), 而收敛步数则大大减少。

- 收敛速度不受网格点数的影响。对于单重网格来说，网格点数越多，其收敛速度越慢，计算量越大；而对多重网格法来说，不管网格点的多少，其收敛速率和所花费的 CPU 时间基本上是一样的。
- 收敛精度高。对于单重网格来说，即使采用双精度，误差也很难下降到 10^{-5} ，否则就不收敛；而对多重网格法来说，误差可下降到 10^{-9} ，并且以相同的速率下降。因此，对于精度要

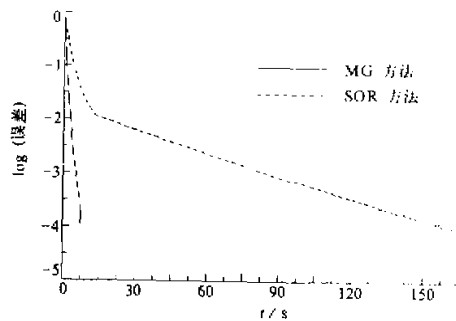


图 3 两种算法的收敛速度

求比较高的问题来说，多重网格法不失为一种值得考虑的方法。

表 1 给出了两种方法达到不同的精度所花费的 CPU 时间以及两种方法的收敛率，在这里收敛率定义为 $|\log(\text{误差})| / \text{工作量}$ 。从表中可见，迭代到一阶和二阶精度时，多重网格法的收敛率分别是 SOR 法的 3.5 倍和 6.24 倍，当误差精度要求为三阶和四阶时，其收敛率之比分别为 16.35 倍和 22 倍。由此可见，采用多重网格法，可在花费较少工作量的情况下达到预定的精确度，它比 SOR 法节省了大量的 CPU 时间。

表 1 两种方法达到不同的精度所花费的 CPU 时间以及收敛率

方法	工作量 (CPU 时间 / s)	精度 (log(误差))	收敛率 ($ \log(\text{误差}) / \text{工作量}$)
SOR	4.124	-1.005	0.244
MG	1.413	-1.223	0.855
SOR	17.88	-2.000	0.112
MG	4.240	-2.668	0.629
SOR	88.96	-3.002	0.034
MG	5.655	-3.121	0.552
SOR	164.4	-4.003	0.024
MG	7.647	-4.002	0.523

4 小结

本文针对迭代法的收敛速度比较慢的缺点，提出了用多重网格法来加速迭代的收敛。因为迭代过程实际上是对误差的光滑化过程。对于固定的网格距迭代对不同频率的误差的光滑程度是不同的。对于高频误差来说迭代的光滑作用大，因此迭代几次就可达到收敛；而对于低频误差，其光滑作用很小，因而需要迭代很多次才能收敛。所以对于单重的网格来说，绝大部分的工作量是用来消除低频误差分量，这是迭代的收敛速度十分缓慢的根本原因。而多重网格法是将网格不断加粗，在每一重网格上迭代以消除一定频率范围内的误差分量，从而达到在整体上消除所有频率的误差分量。这样做的结果大大加快了迭代的收敛速度，并且由于每一重网格的格点数只是上一重网格的 $1/4$ （对

二维问题而言), 其计算量的增加也是很小的。因此, 多重网格法具有收敛速度快, 计算量小的特点。

本文介绍了多重网格法加速迭代收敛的原理, 并且给出了对于线性方程的 MG 算法。利用 MG 算法求解了自适应网格的坐标变换方程, 并与 SOR 迭代法进行了比较。从效果上看, 对于同样的收敛标准, 多重网格法要比 SOR 法节省 20 多倍的 CPU 时间。

参 考 文 献

- 1 Liu Zhuo and Zeng Qingcun, 1992, The application of adaptive mesh to the numerical forecasting of tropical cyclone track, *Proceedings of ICSU / WMO International Symposium*, Edited by James Lighthill et al., Peking University Press, 258~263.
- 2 刘卓、曾庆存, 自适应网格在大气海洋问题中的初步应用, *大气科学*, 18(1), 641~648.
- 3 刘卓、曾庆存, 1995, 完全保持能量守恒的非常定自适应网格模式, *大气科学*, 19(2), 129~139.
- 4 刘卓、曾庆存, 1995, 非常定自适应网格模式在台风路径数值预报中的应用, *大气科学*, 19(3), 258~266.
- 5 刘卓, 1995, 自适应网格在计算地球流体力学中的应用及迭代加速收敛技术——多重网格法, 数值天气预报新技术讲义, 廖洞贤、柳崇健主编, 北京: 气象出版社, 95~124.
- 6 Hackbusch, W. and U. Trottenberg (eds), 1982, *Multigrid Methods, Lecture Notes in Mathematics*, 960, Springer-Verlag.
- 7 Brandt, Achi, 1977, Multi-level adaptive solutions to boundary-value problems, *Mathematics of Computation*, 31, 333~390.

The Application of Multigrid Method in the Adaptive Mesh of Acceleration Generation

Liu Zhuo and Zeng Qingcun

(Laboratory of Numerical Modeling for Atmospheric Science and Geophysical Fluid Dynamics (LASG), Institute of Atmospheric Physics, Chinese Academy of Sciences, Beijing 100080)

Zhang Linbo

(State Key Laboratory of Science and Engineer Computation, Chinese Academy of Sciences, Beijing 100080)

Abstract The multigrid method which was developed in the 1970s is a powerful tool to accelerate the convergence of iterative. It uses different mesh density to eliminate the different frequency component of errors, which accelerate the convergence of iterative. It possess of the advantages with fast convergent speed, high convergent accurate and saving CPU. Multigrid method was applied to accelerate the generation of adaptive mesh in this paper. Experiments show that the multigrid method raises efficiency with more than twenty times compared with single grid.

Key words multigrid method generation of adaptive mesh acceleration of iterative
convergence prediction