文章编号: 1007- 2985(2011) 01- 0026- 04

对溶质运移模型的并行化处理*

邓永辉
(湖南财政经济学院, 湖南 长沙 410205)

摘 要: 在了解和广泛收集前人研究资料的基础上, 将并行算法引入到求解首采区卤水动态二维模型中关于溶质运移的问题中. 把溶质运移方程按时间分裂法分成 5 个子方程, 针对这 5 个子方程讨论了全域精细积分和子域精细积分的并行算法, 给出了对流和扩散方程的子域精细积分并行算法. 子域精细积分考虑了精细积分法高精度的特点, 又避免了全域积分的大矩阵运算, 其精度优于单点精细积分法.

关键词: 对流扩散方程; 并行算法; 偏微分方程
中图分类号: O29 文献标志码: A

溶质运移模型的数值求解法主要包括有限元和有限差分法 2 种. 由于有限差分法计算简单, 物理意义直观, 在溶质运移问题求解中得到普遍应用. 但是, 当对流项占优, 即 Peclet 数较高时, 则出现数值波动和弥散. 采用高阶的时间导数和空间导数离散逼近, 并用极短的时间和空间步长求解, 上游加权法可同时消除大部分的数值弥散, 但求解效率却大大降低. 总之, 对差分法的改进, 或消除了数值波动, 或减少了数值弥散, 仍然未能从根本上消除数值困难. 与有限差分法相比, 有限单元法比较灵活[1], 求解精度高, 因而成为早期求解溶质运移问题应用最广泛的数值法.

1 溶质运移数学模型

通过几种简单模型的推导和分析, 再结合首采区能引起浓度变化的各种因素, 如对流、扩散、排泄、补给等等, 可以导出最终目标模型如下:

$$M(\mu\frac{\partial C}{\partial t}+V_1\frac{\partial C}{\partial x}+V_2\frac{\partial C}{\partial y})=M\mu(\frac{\partial C}{\partial x}(D_{11}\frac{\partial C}{\partial x})+\frac{\partial}{\partial y}(D_{22}\frac{\partial C}{\partial y}))+Q(C_q-C)+M\mu f c. \quad (1)$$

(1) 式只表明了浓度随时间变化的规律, 要求出微分方程的解还需要一些定解条件, 求出在特定条件下浓度的值, 再计算区内边界条件如下:

$$C(x,y,t)\mid_{t=0}=C_0(x,y),\ C(x,y,t)\mid_{\Gamma_1}=C_1(x,y),$$

$$M\mu D_n\frac{\partial C}{\partial n}\mid_{\Gamma_n}=q_{c_2}(x,y),\ \Gamma_1\cup\Gamma_2=\partial\Omega,\ \Gamma_1\cap\Gamma_2=\varnothing.$$

其中: C 为浓度(mL^{-3}); V_1, V_2 为渗透速度(LT^{-1}); D_{11}, D_{22} 为弥散系数(L^2T^{-1}); C_q 为汇源项的浓度(mL^{-3}); n为外法线方向; f 为固液转化系数(T^{-1}), 若忽略固液转化作用, 则$f=0$; $C_0(x,y)$ 为初始浓度分布函数(mL^{-3}); $C_1(x,y)$ 为第 1 类边界 Γ_1 上的浓度分布函数(mL^{-3}); $q_{c_2}(x,y)$ 为第 2 类边界 Γ_2 上弥散通量分布函数($mT^{-1}L^{-1}$); Ω为渗流区; M 为含水层厚度(L); Q 为汇源补给项(LT^{-1}); μ自由孔隙度率, 即给水度, 也就是从单位体积含水层中在重力作用下能够释放给出的水量所占该含水层体积的份数.

* 收稿日期: 2010- 11- 24
基金项目: 国家科技攻关项目(2001BA60ZB- 09- 1)
作者简介: 邓永辉(1979-), 女, 湖南常德人, 湖南财政经济学院讲师, 硕士, 主要从事应用数学领域研究.

2　并行前的预处理

多维的对流扩散方程, 尤其当它为非线性和非定常的时候很难直接求解, 而且在差分格式的选择上还有很大的局限性, 即使得出结果, 解的精度也不高. 为了剔除这些不利因素, Yanenko 提出了一种利用时间分裂的概念[2], 分步求解多维问题的方法. 所谓时间分裂, 笔者是这样理解的, 即在某一固定的时间段内, 一个物理或是化学过程中的若干反应所引起的效力, 并不是同时完成的, 而是存在主次先后之分. 换句话说, 在某一固定的时间段里, 可以将一个物理或化学过程的每个子反应看成是互步相交的集合, 而它们的并形成了这个时间段内完整的物理或化学过程的演变. 因此, 任意多维的问题都可以分解成若干个一维的偏微分方程, 依次求解.

对于方程(1), 不步考虑排泄补给项和溶质的化学转换等, 方程化为

$$(\mu\frac{\partial C}{\partial t}+V_1\frac{\partial C}{\partial x}+V_2\frac{\partial C}{\partial y})=\mu(\frac{\partial C}{\partial x}(D_{11}\frac{\partial C}{\partial x})+\frac{\partial}{\partial y}(D_{22}\frac{\partial C}{\partial y}))+S. \tag{2}$$

根据分步解法, 方程(2) 可以分裂成 5 个方程:

$$\frac{\partial C}{\partial t}=\sum_{r=1}^{R}Q_r\delta(z-z_r)\delta(y-y_r)\delta(x-x_r), \tag{3}$$

$$\frac{\partial C}{\partial t}=-\frac{V_1}{\mu}\frac{\partial C}{\partial x}, \tag{4}$$

$$\frac{\partial C}{\partial t}=-\frac{V_2}{\mu}\frac{\partial C}{\partial x}, \tag{5}$$

$$\frac{\partial C}{\partial t}=D_{11}\frac{\partial^2 C}{\partial x^2}, \tag{6}$$

$$\frac{\partial C}{\partial t}=D_{11}\frac{\partial^2 C}{\partial y^2}. \tag{7}$$

其中: $\delta(\ )$ 为 Dirac 函数; Q_r 为第 r 个源头的源强; x_r, y_r 和 z_r 为这个源的坐标. 方程(3) 为 R 个源头对浓局度部变化$\frac{\partial C}{\partial t}$ 的贡献, (4) 和(5) 式为对流项对浓度局部变化$\frac{\partial C}{\partial t}$ 的贡献, 而(6) 和(7) 式为水平方向的扩散项对浓度局部变化$\frac{\partial C}{\partial t}$ 的贡献.

在对对流扩散方程采用时间分裂法, 将多维的问题转换为 5 个一维的方程求解时, 很大程度上增强了已知算法的可操作性. 下面来研究一维扩散方程和一维对流扩散方程的并行化处理.

3　并行处理

偏微分方程的精细时程积分法先在空间坐标上按差分法离散[3], 而得到对时间的常微分方程组, 然后采用全域精细积分法计算此方程组, 可得到方程组的高精度解. 这相当于给出了偏微分方程的半解析解. 精细积分中指数矩阵 $\boldsymbol{T}$ 的计算量很大, 而 $\boldsymbol{T}$ 的求解主要是矩阵乘法的运算. 全域精细积分中, 需要对矩阵运算进行并行化. 文中使用的 Transputer 系统是分布式存储的多处理器系统, 采用分块的矩阵乘法. 变量较多时, 全域精细积分的指数矩阵的存储量和计算量都很庞大. 子域精细积分不需形成总体的指数矩阵, 各子域的求解是相对独立的, 计算量较大的指数矩阵求解可在各处理器中分别计算, 它具有良好的并行计算的性质.

3.1 一维扩散方程的并行算法

(1) 全域精细积分的并行算法.

考虑一维扩散方程$\frac{\partial C}{\partial t}=D\frac{\partial^2 C}{\partial x^2}$, 当两端边界条件为 0, 用差分法在 x 坐标方向离散后, 可得到

$$\mu^{k+1}=\exp(\boldsymbol{H}\cdot t)\cdot\mu^0=\exp(\boldsymbol{H}\cdot\Delta t)\cdot\mu^k=\boldsymbol{T}\cdot\mu^k, \tag{8}$$

其中 $\boldsymbol{H}$ 是常数矩阵, μ^0 是初值. 文献[4] 中给出了指数矩阵 $\boldsymbol{T}$ 的精细计算方法:

$$\boldsymbol{T}=\exp(\boldsymbol{H}\cdot\Delta t)=[\exp(\boldsymbol{H}\cdot\Delta t/m)]^m=[\exp(\boldsymbol{H}\cdot\boldsymbol{T})]^m=[\boldsymbol{I}+\boldsymbol{T}_a]^m,$$

其中可选用 $m=2^N$. 若 Δt 是一个大的时间区段, 则当 N 并不很大时, $\tau=\Delta t/m$ 也将是一个非常小的时间

段. $\boldsymbol{T}$ 的计算相当于做 N 次循环:

$$\boldsymbol{T}_a = 2\boldsymbol{T}_a + \boldsymbol{T}_a \times \boldsymbol{T}_a. \tag{9}$$

最后做 $\boldsymbol{T} = \boldsymbol{I} + \boldsymbol{T}_a$, 这样可得到高精度的指数矩阵 $\boldsymbol{T} = \exp(\boldsymbol{H} \cdot \Delta t)$.

Transputer 系统是分布式存储的并行多处理器系统, 各处理器的数据由相互间的通讯得到. Fortran 语言中数组的数据是按列存放的. 为了通讯的方便和高效率, (9) 式中矩阵的计算按列分块 $\boldsymbol{T}_a = (\boldsymbol{T}_{a1}, \boldsymbol{T}_{a2}, \cdots, \boldsymbol{T}_{an})$. n 是处理器的数目. 处理器 i 中的矩阵计算为 $\boldsymbol{T}_{ai} = 2\boldsymbol{T}_{ai} + \boldsymbol{T}_a \times \boldsymbol{T}_{ai}$. 由于相乘和相加是相同的 $\boldsymbol{T}_a$ 矩阵, 因此计算时先将 $\boldsymbol{T}_a$ 阵分别送到各个处理器, 各处理器计算 $\boldsymbol{T}_{ai}$; 然后进行通讯得到新的完整的 $\boldsymbol{T}_a$ 矩阵; 再将 $\boldsymbol{T}_a$ 阵送给各处理器, 依次循环后, 加上单位阵 $\boldsymbol{I}$ 得最终的矩阵 $\boldsymbol{T}$. (8) 式的时间步计算中, 将 $\boldsymbol{T}$ 阵按行分块 $\boldsymbol{T} = (\boldsymbol{T}_1, \boldsymbol{T}_2, \cdots, \boldsymbol{T}_n)$, μ^{k+1} 则分为 n 段 $\mu^{k+1} = ((\mu_1^{k+1})'(\mu_2^{k+1})' \cdots (\mu_n^{k+1})')'$, $\mu_i^{k+1} = \boldsymbol{T}_i \times \mu^k$, $i = 1, 2, \cdots, n$. 由通讯收集得到 μ^{k+1}, 再送给各处理器进行下一时间段的计算.

(2) 子域精细积分的并行算法.

当未知数增多时, 全域精细积分法指数矩阵的存储量和计算量都将迅速增加, 这会影响到其解题的规模和效率. 当时间步长 Δt 很小时, 指数矩阵 $\boldsymbol{T}$ 是近似窄带宽的. 因此, 文献[5] 提出了子域精细积分的方法, 这可以使存储量和计算量大幅度下降.

考虑子域的非齐次微分方程组, 其向前精细积分格式为

$$v_j^{n+1} = \boldsymbol{T} v_j^n + (\boldsymbol{T} - \boldsymbol{I})\boldsymbol{H}^{-1} f_j^n \qquad j = 1, 2, \cdots, J. \tag{10}$$

蛙跳格式为

$$\boldsymbol{T}(2\Delta t) = \exp(2\boldsymbol{H} \cdot \Delta t),$$

$$v_j^{n-1} = \boldsymbol{T} v_j^n + (\boldsymbol{T} - \boldsymbol{I})\boldsymbol{H}^{-1} f_j^n \qquad j = 1, 2, \cdots, J. \tag{11}$$

格式(10) 是一阶时间精度, 格式(11) 则是二阶时间精度, 而两者的计算量几乎是一样的. 当各子域划分相同时, 只需计算 1 组 $\boldsymbol{T}$ 和 $\boldsymbol{H}^{-1}$ 阵. T 阵的并行算法与全域精细积分的相同, $\boldsymbol{H}^{-1}$ 阵的计算采用分块的 Gauss-Jordon 算法. 将 $\boldsymbol{H}$ 按列分块送到各处理器:

$$\boldsymbol{H} = (\boldsymbol{H}^1, \boldsymbol{H}^2, \cdots, \boldsymbol{H}^n).$$

设有 m 个内点. 从 $\boldsymbol{H}$ 阵的第 1 列到最后的第 m 列依次计算. 计算第 j 列时, j 列所在的第 I 个处理器从第 j 列选主元. 第 I 个处理器将选出的主元行号 k 及第 j 列向量 $\boldsymbol{a}$ 送到每个处理器. 各处理器交换 $\boldsymbol{H}$ 阵中的第 j, k 2 行:

$$tmp = \boldsymbol{H}_{jl}^i, \boldsymbol{H}_{jl}^i = \boldsymbol{H}_{kl}^i, \boldsymbol{H}_{kl}^i = tmp \qquad l = 1, \cdots, m; i = 1, \cdots, n.$$

选出的主元行号存储于一维数组 M 中. 然后进行消去运算. 设 $\boldsymbol{H}$ 阵的第 j 列在 $\boldsymbol{H}_i$ 阵中为第 p 列, 则第 I 个处理器中的算法为:

$$\begin{cases} \boldsymbol{H}_{jp}^I = 1/a_j, \boldsymbol{H}_{lp}^I = -a_l/a_j & l = 1, \cdots, j-1, j+1, \cdots, m; \\ \boldsymbol{H}_{jl}^I = \boldsymbol{H}_{jl}^I/a_j & l = 1, \cdots, p-1, p+1, \cdots, m_I; \\ \boldsymbol{H}_{ql}^I = \boldsymbol{H}_{ql}^I - \boldsymbol{H}_{jl}^I a_q & q = 1, \cdots, j-1, j+1, \cdots, m, l = 1, \cdots, p-1, p+1, \cdots, m_I. \end{cases}$$

其他处理器中的算法为:

$$\begin{cases} \boldsymbol{H}_{jl}^I = \boldsymbol{H}_{jl}^I/a_j & l = 1, \cdots, m_i, i = 1, \cdots, n; \\ \boldsymbol{H}_{ql}^I = \boldsymbol{H}_{ql}^I - \boldsymbol{H}_{jl}^I a_q & q = 1, \cdots, j-1, j+1, \cdots, m, l = 1, \cdots, m_i, i = 1, \cdots, n. \end{cases}$$

$\boldsymbol{H}$ 阵是按列存在各处理器中, 交换的通讯量很大, 可将各处理器中的 $\boldsymbol{H}_i$ 送到主控程序[4], 由主控程序进行最后的列交换, 得到逆阵 $\boldsymbol{H}^{-1}$. 再将 $\boldsymbol{T}$ 和 $\boldsymbol{H}^{-1}$ 阵送回各处理器. 向量 μ 按照子域划分情况送到各处理器, 按(10) 或(11) 式计算各子域的 v_j^{n+1}. 将 v_j^{n+1} 中与其他子域重合部分的分量送出, 同时接收其他处理器计算的这几个分量, 并取平均值作为计算结果.

3.2 对流扩散方程的并行算法

考虑更一般的一维对流扩散方程

$$\frac{\partial u}{\partial t} = D \cdot \frac{\partial^2 u}{\partial x^2} - v \cdot \frac{\partial u}{\partial x}, \tag{12}$$

为改进差分法的数值精度和稳定性, 文献[4] 给出了对流扩散方程的单点精细积分算法. 由方程 (12) 的 2

个通解 $u = $ 常数和 $u = x - vt$, 得到在空间坐标上离散的微分方程[5].

$$u_j^0 = c_1 u_{j-1} - c_2 u_j + c_3 u_{j+1}. \tag{13}$$

(13) 式的单点蛙跳格式为

$$u_j^{n+1} = [u_j^{n-1} - (c_3 u_{j+1}^n + c_1 u_{j-1}^n)/c_2] \cdot \exp(-2c_2 \cdot \Delta t) + (c_3 u_{j+1}^n + c_1 u_{j-1}^n)/c_2. \tag{14}$$

此格式具有二阶时间精度, 并且是无条件稳定的. 为提高精度, 可以采用多内点的子域精细积分算法. 利用蛙跳格式(14), 考虑 3 个内点的子域, 其非齐次微分方程组是

$$\begin{Bmatrix} \overset{0}{u}_{j-1} \\ \overset{0}{u}_j \\ \overset{0}{u}_{j+1} \end{Bmatrix} = \begin{pmatrix} -c_2 & c_3 & 0 \\ c_1 & -c_2 & c_3 \\ 0 & c_1 & -c_2 \end{pmatrix} \begin{Bmatrix} u_{j-1} \\ u_j \\ j_{j+1} \end{Bmatrix} + \begin{Bmatrix} c_2 u_{j-2} \\ 0 \\ c_3 u_{j+2} \end{Bmatrix} \qquad j = 1, 2, \cdots, J. \tag{15}$$

u_{j-2} 和 u_{j+2} 作为常数, 在 Δt 时间段内是不变的, 可以从相邻子域中得到这 2 点的值. 采用与(10), (11) 式相同的格式计算方程(15).

当在空间坐标作等距离的离散, 并且各子域内点相同时, $\boldsymbol{T}_j$ 和 $\boldsymbol{H}_j^{-1}$ 是相同的, 则只需计算 1 组[6]; 算法与前相同. 使用不同划分的子域时, 先将向量 $\boldsymbol{x}$ 按划分的子域分段地送到各处理器, 各处理器计算各系数, 并组装得到 $\boldsymbol{H}_j$ 矩阵; 然后计算 $\boldsymbol{H}_j^{-1}$ 和 $\boldsymbol{T}_j$. 这一步计算量较大, 且不需通讯, 是大粒度的.

4　结语

精细积分法有很高的计算精度, 不论是全域积分还是子域积分都有很好的并行性, 在并行机系统上可充分发挥其效能. 其中子域积分法有更大的优点, 它克服了全域积分矩阵过大的弱点, 又可以提高计算的精度. 在并行计算中能方便地将各子域放到各处理器中, 算例表明此算法是高效率的. 增加内点数加大了指数矩阵和每个时间步的计算量, 但能够提高精度, 所以应适当地选择子域的大小.

参考文献:

[1] 王文科. 地下水有限分析数值模拟的理论与方法 [M]. 西安: 陕西科学技术出版社, 1996: 102- 126.

[2] ROACHE P J. Computational Fluid Dynamics [M]. Hermosa, Albuquerque, New Mexico, 1976: 446.

[3] 孙纳正. 地下水污染——数学模型和数值方法 [M]. 北京: 北京地质出版社, 1989.

[4] 钟万勰. 子域精细积分及偏微分方程数值解 [J]. 计算结构力学及其应用, 1995, 12(3): 253- 260.

[5] 罗焕炎, 陈雨孙. 地下水运动的数值模拟 [M]. 北京: 中国建筑工业出版社, 2001.

[6] DURBIN F. Numerical Inversion of Laplace Transform: An Efficient Improvement to Durbner and Abare's Method [J]. Comp. J, 1973, 17: 371- 376.

Parallel Algorithms for a Solute Transport Model

DENG Yong-hui

(Hunan University of Finance and Economics, Changsha 410205, China)

Abstract: This paper presents the parallel algorithms of precise integration methods. The subdomain parallel algorithms of precise integration of convection-diffusion equations are given. The method take advantage of the high precision of precise integration methods, and they also avoid a large amount of matrices calculation of whole domain methods. The precision of the methods is better than that of the single point integration.

Key words: convection-diffusion equations; parallel algorithm; partial differential equations

(责任编辑　向阳洁)