维普资讯 http://www.cqvip.com

常微分方程 并行计算，初值问题
数值计算

④

212-224

常微分方程的并行计算方法

Kevin Burrage

O241.81

Burra., K

在并行处理中，常常注重偏微分方程 (PDE)，而忽略常微分方程 (ODE). 本文介绍常微分方程的并行计算方法，这些算法不仅能保持计算精度，而且既适合于 SIMD 并行机，也适合于 MIMD 并行机.

近年来，对常微分方程初值问题

$$y' = f(t,y),\ y(t_0) = y_0 \quad f : \mathbb{R} \times \mathbb{R}^m \Rightarrow \mathbb{R}^m, \tag{1}$$

有效的并行数值方法进行了相当的研究. 这里大气污染物长距离迁移的数学模型 [14] 就是一例，它给出了有关问题的许多思想. 为研究污染物季节性变化，一个较简单的数学模型就包含有 267,264 个常微分方程，而且需在长时间尺度上求解. 很明显，若不开发并行算法，这样大规模的问题不可能在适度的时间内计算. 在求解问题 (1) 时，已确认有三种类型的并行性：

(1) 方法的并行性；

(2) 方程组 (空间) 的并行性；

(3) 时间的并行性.

有效的并行算法很可能同时考虑这三种并行因素，使这些算法处于三维空间中 (图 1)

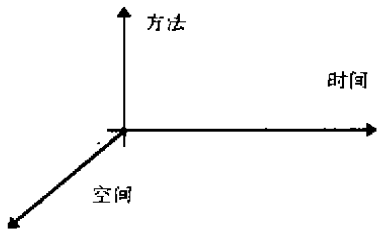

图 1. 并行空间

本文考虑两种算法，且给出数值结果以说明这些方法的效率. 一种方法是基于方法的并行，它适合于在 SIMD(单指令多数据流) 并行机上实现. 如本文的工作是在 MasPar 机上实现的. 另一种算法是基于方程组的并行性. 它适合于在 MIMD(多指令多数据流) 并行机上实现，如 iPSC860.

原题: Parallel Methods for ODEs. 译自: *SIAM NEWS*, Vol. 26, No. 5, 1993.

维普资讯 http://www.cqvip.com

方法的并行性

方法的并行性的一种开发途径是在不同的处理器上同时计算几个函数，例如，多级 Runge–Kutta 法就可以用几个处理器同时计算每级右端的函数值．一般说来，这在直接方法中没有多大好处．不过，对规模大的问题或函数计算很耗机时的情况下，外推技术很适合于并行计算，它用处理器来均衡计算量 [5]．另外，可以证明如预测－校正法等间接法在并行计算中也是很有效的．

预测－校正法

方法并行性的普通开发方法 [11] 是基于块方法 (block method) 概念，即用某种显式方法从前面已算出的一组值并行地预测一块值，再用隐式方法进行若干次不动点迭代去校正预测值，例如，为了同时更新等距节点 $t_{n+1}\cdots t_{n+k}$ 上 k 个值，通常采用显示 Euler 预测器，再用梯形校正器校正二次，即如下三步：

$$\begin{aligned} y_{n+j}^{(0)} &= y_n + jhf(t_n, y_n), & j &= 1,2,\ldots,k \\ y_{n+j}^{(1)} &= y_n + \frac{h}{2}f(t_n, y_n) + \frac{h}{2}f(t_{n+j}, y_{n+j}^{(0)}), & j &= 1,2,\ldots,k \\ y_{n+j}^{(2)} &= y_n + \frac{h}{2}f(t_n, y_n) + \frac{h}{2}f(t_{n+j}, y_{n+j}^{(1)}), & j &= 1,2,\ldots,k. \end{aligned} \tag{2}$$

虽说该方法很简单，但它是一种很通用的方法，即基于一个 Hermite 预测器：

$$y^{(0)} = A_0 \otimes Y_n + hL_0 \otimes F(y_n) \tag{3}$$

和一个隐式校正器：

$$Y^{(j)} = Z_n + hL_2 \otimes F(Y^{(j-1)}), \quad j = 1,2,\ldots,r, \tag{4}$$

$$Z_n = A_1 \otimes Y_n + hL_1 \otimes F(Y_n),$$

其中 $F(y_n)$ 表示一个向量，其分量为 $f(y_1^{(n)}),\cdots,f(y_k^{(n)})$.

一般说，如果不作多次校正的话，这种方法的稳定性很差，和 / 或有大的误差系数 [3]．通常，每作一次形如 (4) 的校正，方法的阶提高一阶，直达到校正器的阶数 [2]．再继续进行校正，方法的阶不会再提高，但可以逐次光滑局部截断误差式中阶数越来越高的截断系数．块越大，预测器中外推误差也越大．需作多次校正才能保证达到预期精度．然而，直接对校正器 (4) 采用分裂技巧 (可看作预处理) 可大大提高其计算效率．

这种分裂方法的一般迭代格式为：

$$M_{k,n}Y_{n+1}^{(k+1)} = M_{k,n} - IY_{n+1}^{(k)} + Z_n + hL_2 \otimes F(Y_{n+1}^{(k)}), \quad k = 0,1,\ldots \tag{5}$$

维普资讯 http://www.cqvip.com

当

$$M_{k,n} = I, \qquad \forall k, n, \tag{6}$$

时，式 (5) 就是标准的预测－校正法. 当

$$M_{k,n} = I - hL_2 \otimes J_n, \quad \forall k, \tag{7}$$

时，正是不动点迭代，其中 J_n 是待计算问题在点 y_n 处的 Jacobi 行列式，则式 (5) 就表示一个修正的 Newton 法.

对式 (6) 和式 (7) 中的 $M_{k,n}$ 作适当选取，使该方法在并行环境中既能达到好的收敛性又能廉价地实现算法.

我们定义

$$e_{n+1}^{(k+1)} = Y_{n+1}^{(k+1)} - Y_{n+1}, \tag{8}$$

则问题线性化后得

$$e_{n+1}^{(k+1)} = R_{k,n} e_{n+1}^{(k)},$$

$$R_{k,n} = I - M_{k,n}^{-1}(I - hL_2 \otimes J_n). \tag{9}$$

另一种方法是把基本校正器 (4) 用于线性问题

$$Y'(t) = J(t)Y, \tag{10}$$

则有

$$PY^{(k)} = Z_n, \qquad k = 1, \ldots, r,$$

$$P = I - hL_2 \otimes J_n. \tag{11}$$

选择式 (6) 和式 (7) 中的 $M_{k,n}$ 对矩阵 P 作预处理将会对迭代公式 (5) 加速. 如果已知问题 (1) 的特征值结构 (比如，用线方法求抛物型偏微分方程得到的常微分方程组就知道其特征值的结构)，则多项式预处理是加速收敛的一种著名的方法，例如

$$\text{I 型}: M_k^{-1} = \alpha I \Rightarrow R_{k,n} = I - \alpha P$$

$$\text{II 型}: M_k^{-1} = \alpha_k I - \beta_k P \Rightarrow R_{k,n} = I - \alpha_k P + \beta_k P^2. \tag{12}$$

若 J_n 的特征值是实的且位于区间 $[-q, 0]$ 内，$q > 0$，当 $\rho\left(\prod_{k=1}^{p} R_{p-k,n}\right)^{\frac{1}{p}}$ 最小时，则 p 次迭代的收敛速度达到最大，其中 $\rho(H)$ 为 H 矩阵的谱半径.

式 (4) 的 Chebyshev 多项式分析表明：若

$$\alpha = \frac{2}{1+v}, \quad \rho(R) = 1 - \alpha$$

$$\alpha = \frac{8(1+v)}{1+6v+v^2}, \ \beta = \frac{\alpha_k}{1+v}, \ \rho(R) = 1 - \alpha + \beta$$

维普资讯 http://www.cqvip.com

$$\alpha_k = \frac{4}{4v + S_k^2(1-v)^2}, \quad \beta_k = \frac{\alpha_k}{1+v},$$

$$S_k^2 = \sin^2\left(\frac{(2k-1)\pi}{4p}\right), \quad k = 1, 2, \ldots, p. \tag{13}$$

其中 $v = det(I + ZL_2)$(对梯形公式, $v = 1 + Z/2$) 时，则放大矩阵 (为 $z = qh$ 的函数) 的谱半径达到最小.

这种方法的优点是计算机上的实现与显示方法类似，而其稳定性又与 A 稳定的隐式方法类似. 而且校正计算仅为简单的矩向量运算，从而很省机时. 另一个优点是不需要求解线性方程组，解线性方程组的算法在并行环境下是难以有效地编程的. 该方法特别适合于 SIMD 计算机，如 MasPar 并行机. 这已在昆士兰 (Queensland) 大学的 4k 处理器 MasPar MPI 上编程.

MasPar

一个阵列机是一大批处理组件 (PEs) 按网格拓扑或某种导数形式排列. 特别地，PEs 是同步运算，即所有的 PEs 同时执行相同的指令和不同的数据. 由于诸如流体力学、应力分析及空间模型中的很多数学模型问题都可用空间离散网格近似，因此有一个自然处理器拓扑，从而利用 FORTRAN90 结构，特别是 BLAS 子程序结构，自动地实现计算并行化 (见下面).

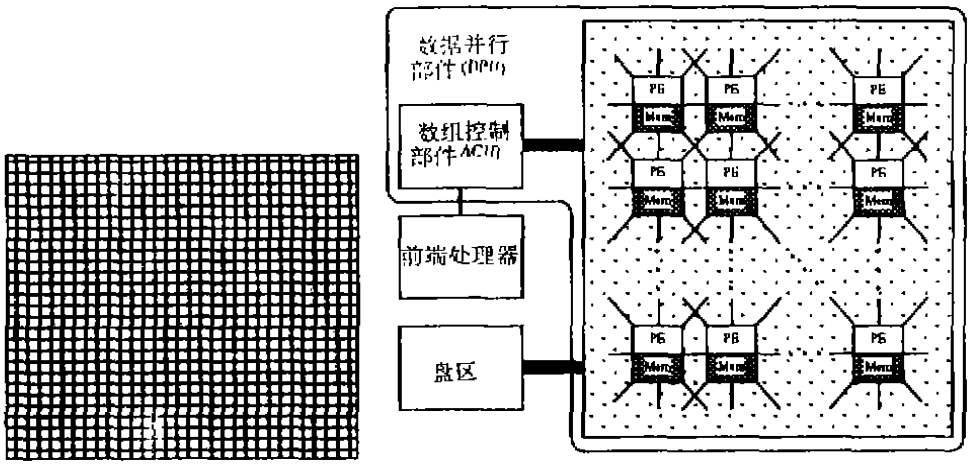

图 2. 计算空间和网络拓扑

昆士兰大学的 MasPar MP1 计算机 (图 2) 是由一个执行串行计算的前端 Unix 工作站及一个后端数据并行部件 (DPU) 组成的，而 DPU 又由一个阵列控制部件 (ACU) 和一组 PE 构成. ACU 处理全部标量程序. 处理组件是由 64,000 字节存贮器的 4 位处理器组成. 每个 PE 可持续执行约 50,000 次浮点操作 (flop).

维普资讯 http://www.cqvip.com

通过 XNET 经过 8 个最邻近的处理器进行通讯. XNET 是一个同步 (lock-step) 内处理器通讯协议 (Protocol). 换句话说，通过一个三级开关发送器，则全局发送器允许任何处理器之间进行通讯.

前节所述的预处理方法已编成 MPFortran 程序， MPFortran 是以新的 Fortran 标准为基础的一种高级语言，由于它只需要 BLAS 型运算， Fortran 编译器自动将数组和向量在 DPU 上分块. 全部优越性正是由于 Fortran 编译器充分利用了并行性.

方程组的并行性

方程组并行性的最简单的开发方法是采用 Picard 迭代概念，更一般地，采用在函数空间中进行迭代的概念. 为了得到式 (1) 解的整体逼近， Picard 方法采用如下形式的函数迭代序列

$$y'^{(k+1)}(t) = f(t, y^{(k)}(t)),$$

$$y^{(k+1)}(t_0) = y^{(k)}(t_0). \tag{14}$$

此时，在每个迭代级上可把问题分裂成 m 个独立的求积问题. 看来这似乎是达到整体并行性的一种合适的方法. 遗憾的是，迭代序列 $\{y^{(k)}(t)\}$ 收敛到 $y(t)$ 的速度非常慢. 例如，对于标准的线性试验问题：

$$y' = \lambda y \quad t \in [0, T], \quad \lambda < 0, \tag{15}$$

可以证明迭代值 $y^{(k)}(t)$ 满足如下的整体误差界：

$$\left|y(t) - y^{(k)}(t)\right| \leq \frac{(|\lambda| t)^{k+1}}{(k+1)!}, \quad t \in [0, T], \tag{16}$$

显然，只有当 $k \geq |\lambda| T$ 时才收敛. 因此，改进收敛性的一种方法是用窗口技术，即将求积区域分裂成一系列的窗口，并在每个窗口上作迭代.

首先对更一般的函数迭代格式作广泛研究的是 California 大学 (Berkeley) 的电子工程研究组在 80 年代初进行的 [9,13], 他们将这种函数迭代格式取名为 "波形松弛". 用这种技巧可将解线性代数方程组的标准迭代格式直接应用于微分方程组，从而得到收敛于式 (1) 解的微分方程组序列，其中每个方程均可用离散方法求解. 因此，下面分别介绍 Jacobi 波形法 - 式 (17) 和 Gauss-Seidel 波形法 - 式 (18):

$$y'^{(k+1)}(t) = f_i\left(y_1^{(k)}, \ldots, y_{i-1}^{(k)}, y_i^{(k)}, y_{i+1}^{(k)}, \ldots, y_m^{(k)}\right), \quad i = 1, \ldots, m \tag{17}$$

$$y'^{(k+1)}(t) = f_i\left(y_1^{(k+1)}, \ldots, y_{i-1}^{[k+1]}, y_i^{(k)}, y_{i+1}^{(k)}, \ldots, y_m^{(k)}\right), \quad i = 1, \ldots, m. \tag{18}$$

对于非线性情况，容易证明式 (17) 和式 (18) 的结果满足式 (16), 但收敛很慢. 然而，对某些类似的问题，比如， Bekerley 研究组研究的积分回路设计问题，这些波形松

维普资讯 http://www.cqvip.com

弛技巧确实很好. 这是由于物理模型本身给出了将组件分成仅在极短的时间间隔内呈现紧耦合系统的方法.

一般说，困难往往出现在如何选择待分组的组件及如何把方程重新排序. 排序方式对波形松弛的效率至关重要 [7]. 可以看出，在子系统强耦合情况下，波形松弛收敛很慢.

近来，多重网格法 (Multigrid) 加速技术已直接应用于线性方法求解抛物型微分方程得到的线性问题. 这些技术确实能大大地加速这种迭代格式的收敛性 [10]. 文献 [12] 已将此工作推广到非线性问题.

分布计算

波形方程适合于在分布环境下实现. 因为分布环境允许把原问题解耦成子问题. 这些子问题可在不同程度上相互独立地在不同处理器上求解 (当然，这依赖于原问题各种成分耦合的性质). 这种方法使编程人员可利用现有的在串行环境下有效和健壮的串行程序. 且可用他们求解一组子问题. VODE[1] 就是这样一种程序. 它以 Adams 法和 BDF 法为基础，既适合于刚性问题也适合于非刚性问题.

分布计算要求编程人员必须把注意力集中在通讯协议上. 这样他们就可以在普通信息传输环境下编程，如 PVM(并行虚拟机)[8] 或 p4(并行处理器上可移植的程序)[6]. 这些软件环境适合于编制关于求具有高度粒状并行性子问题的 Fortran77 子程序或 C 子程序，软件环境 PVM 和 p4 都是基于信息传输模型 (允许信息传输) 、障碍同步及广播. PVM 和 p4 之间的主要差别是： PVM 用一个 pvmd 智能程序 (demon) 去控制过程状态，而 p4 却没有用. p4 是通过自然信息传输在分布存储器上通讯，而 PVM 用智能程序.

已用 VODE 作为基本积分器及 p4 作为信息传输 "沾结剂"(glue) 编制了程序. 数值结果见下节. p4 的优点是能够调试 (debug) 程序，在工作站网上测试以及不用作任何改变很容易转向 iPSC860.

数值结果

为了说明前述的一些内容，计算了两个二维偏微分方程试验问题. 第一个问题是定义在单位正方形上具有 Dirichlet 边界条件的线性扩散方程

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \tag{19}$$

$$u(t,x,0) = u(t,x,1) = u(t,0,y) = u(t,1,y) = 1, \tag{19}$$

用线性方法把它变成一组常微分方程. 取网格步长 $h = 1/N + 1$, 在此均匀网格上用中心差分去离散二阶空间导数，得 N^2 个线性微分方程组

$$u' = (N+1)^2 Qu,$$

维普资讯 http://www.cqvip.com

$$u(0) = 1. \tag{20}$$

其中 Q 为块三对角阵 (I_N, T, I_N), I_N 为 N 阶单位阵, T 是对角线元素为 -4, 上下次对角线元素为 1 的三对角阵. 对这个问题, q 等于 $8(N+1)^2$.

第二个问题是一个反应 - 扩散方程, 称为 Brusselator 扩散方程 [12], 即

$$\begin{cases} \dfrac{\partial u}{\partial t} = B + u^2 v - (A+1)u + \alpha\left(\dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial x^2}\dfrac{\partial v}{\partial t}\right), \\ \quad = Au - u^2 v + \alpha\left(\dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 v}{\partial y}\right), \end{cases} \tag{21}$$

初值条件为

$$u(0,x,y) = 2 + 0.25y \qquad v(0,x,y) = 1 + 0.8x,$$

Neumann 边值条件为

$$\frac{\partial u}{\partial n} = 0, \quad \frac{\partial v}{\partial n} = 0,$$

其中 $A = 0.34, B = 1$ 和 $\alpha = 0.002$. u 和 v 表示反应产物的化学浓度. A 和 B 是输入试剂的常值浓度. α 是一个与扩散系数和反应器长度有关的常数.

用中心差分去离散式 (21) 中的二阶空间导数, 得到 $2(N+2)^2$ 个耦合的非线性方程组

$$\begin{aligned} u'_{i,j} &= B + u_{i,j}^2 v_{i,j} - (A+1)u_{i,j} + \hat{\alpha}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}) \\ v'_{i,j} &= Au_{i,j} - u_{i,j}^2 v_{i,j} + \hat{\alpha}(v_{i+1,j} + v_{i-1,j} + v_{i,j+1} + v_{i,j-1} - 4v_{i,j}), \end{aligned} \tag{22}$$

其中 $\hat{\alpha} = \alpha(N+1)^2$.

SIMD 实现

在 4k MasPar 机上用定步长梯形校正器求解了一个线性扩散方程和一维 Brusselator 方程. 计算结果如下:

1. 当计算 N^2 维线性问题时, 必须小心地选取 N. 若取 $N = 65$, 所用处理器的数目是 $64 \times 64 = 4096$. 计算时间约是 $N = 64$ 的二倍. 原因是 MasPar 自动将计算网格分层给存储器, $N = 65$ 时需要二层. 这种分层是自动进行的, 不需要编程人员考虑. 另一方面, 尽管计算时间与计算机负荷有关, 但若取 $N \le 64$, 则求解 N^2 维问题所需的时间应当近似相同.

2. 虽然如方法的并行性一节所述: 计算机上的实现只需用运算 Qv 的 BLAS, 其 Q 与式 (20) 中的一样, v 是一个向量, 由计算网格的全部元素构成. 但重要的是要把问题构造和使运算非常有效. 于是, 把 v 表示成 $N \times N$ 矩阵, 把 Qv 构成为 EOSHIFT 序列:

维普资讯 http://www.cqvip.com

EOSHIFT(v,SHIFT=-1, BOUNDARY=f1, DIM=1)+
EOSHIFT(v,SHIFT=-1, BOUNDARY=f2, DIM=2)-4.0*v^+
EOSHIFT(v,SHIFT=+1, BOUNDARY=f3, DIM=2)+
EOSHIFT(v,SHIFT=+1, BOUNDARY=f4, DIM=1).

其中 SHIFT 表示上移下移计算网格，DIM 表示行和列移位，$f1, f2, f3$ 和 $f4$ 是边界条件.

3. 对一维 Brusselator 方程，两个耦合的 N 维向量自动分层为 MasPar 拓扑上的两个行向量. 对 II 型实现 (见方程 (12)), 每个时间步需计算一个 Jacobi 矩阵. 因为 Jacobi 矩阵有一个简单的三对角块结构，其中非对角块为单位阵，Jacobi 阵与表示问题组分的每个向量的乘积很容易构成两个顺列方向的 EOSHIFT 序列.

4. 方程 (20) 也已用大小为 2 的块方法求解. 该方法采用二级三阶 RADAU 校正器. 这时，每个处理器计算两个近似值 (一个在积分步长的 1/3 处，另一个在积分步长的末端), 计算时间与预期的一样，差不多是梯形校正器的二倍.

MIMD 实现

在 Sparc-2 工作站的分布线上及在德国茹利希 (Jülich) 的具有 32 个处理器的 iPS860 上，采用 Jacobi 波形松弛及 ODE 软件包 VODE, 求解了二维 Brasselator 方程. 所有的自动时间窗口是基于迭代收敛的自适应控制. 在 p4 中编制了信息传输程序，把二维 Brasselator 问题在单个处理器上 VODE 串行运行与用 32 个处理器的波形运行作了比较. 图 3 显示了增速随维数 ($2N^2$) 变化的两个图. 一个图表示各分量之间不并行 (Overlap), 另一个是最优并行 (对一个特定的维数用最佳增速). 一个 800 维问题在一台 32 节点计算机上增速约 7 倍是令人满意的.

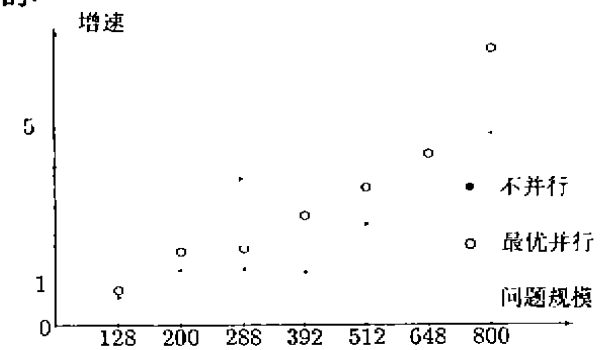

图 3. 增速随维数的变化： [0,6]; 80 个时间点； 32 个处理器

结论

MasPar 所得结果指出，求解刚性大的问题在每个时间步可能不需要去求解大型线性方程组. 而且这些技巧很适合于大型并行计算机.

维普资讯 http://www.cqvip.com

在 MIMD 环境下，本文重点介绍的方法指出，并行算法的研究应当尽可能地利用现有的既健壮又有效的串行软件包. 这样不仅使并行算法具有健壮性，而且也将使编程人员可把注意力集中在处理器之间的通讯上. 使用 PVM 或 p4 这样的软件包将会带来很大的方便. 最后，用多重网格波形法可以改进算法在计算机上的实现.

有限的篇幅不可能更详细地讨论更广泛类型的常微分方程并行算法. 显然，各种算法不会在结构不同的计算机上同样运行得很好. 不久的将来，既依赖于问题又依赖于计算机的结构的各种并行程序必定多于串行程序. 而且，并行 MIMD 计算机用于自动售货机已有明显的趋势，很可能这种情况是暂时的，其他领域今后也可能会大量使用. 我们预期，不仅在微分方程领域，而且在计算机领域，大量的并行计算机上的程序会有很好的一致性和可移植性.

感谢

作者感谢 Pamela Burrage 和 Bert Pohl 分别在 MasPar 和 iPS860 上编程给予的帮助.

参 考 文 献

[1] P.N.Brown, G.D.Byrne, and A.C.Hindmarsh, *VODE: A variable-coefficient ODE solver*, SIAMJ. Sci. Stat. Comp., **20** (1989), 1038–1051.

[2] K. Burrage, *The error behavior of a general class of predictor—corrector methods*, App. Num. Math., **8** (1991), 201–216.

[3] K. Burrage, *Parallel methods for initial value problems*, App. Num. Math., **11** (1991), 5–25.

[4] K. Burrage, *Parallel and sequential methods for differential equations*, in press, Oxford University Press.

[5] K. Burrage and S. Plowman, *The numerical solution of ODEIVPs in a transputer environment*, Applications of Transputers 2, D. J. Pritchard, C. J. Scott, eds., IOS Press, 1990, 495–505.

[6] R. Butler and E. Lusk, *User's Guide to the p4 Programming System*, Report ANL -92/17, Argonne National Laboratory.

[7] C. H. Carlin and C. Vachoux, *On partitioning for waveform relaxation time-domain analysis of VLSI circuits*, in Proc. of Int. Conf. on Circ. and Syst., Montreal, 1984.

[8] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderan, *PVM 3.0 User's Guide and reference Manual*, Report ORNL/TM—12187, Mathematical Sciences Section, Oak Ridge National Laboratory.

[9] E. Lelarasmee, *The waveform relaxation method for the time domain analysis of large scale nonlinear dynamical systems*, PhD thesis, University of California, Berkeley, CA, 1982.

[10] C. Lubich and A. Ostermann, *Multigrid dynamic iteration for parabolic equations*, BIT, **27** (1987),

维普资讯 http://www.cqvip.com

216–234.

[11] W. L. Miranker and W. Liniger, *Parallel methods for the numerical integration of ordinary defferential equations*, Math. Comp., **21** (1967), 303–320.

[12] S. Vandewalle and R. Piessens, *Numerical experiments with nonlinear multigrid waveform relaxation on a parallel processor*, App. Num. Math., **8** (1991), 149–161.

[13] J. White, A. Sangiovanni-Vincentelli, F. Odeh, and A. Ruchli, *Waveform relaxation: Theory and practice*, Trans. of Soc. for Computer Simulation, **2** (1985), 95–133.

[14] Z. Zlatev, *Treatment of some mathematical models describing long-range transport of air pollutants on vector processors*, Parallel Computing, **6** (1988), 87–98.

(顾丽珍 译　王 铎 校)

(上接 211 页)

[3] N. Seiberg and E. Witten, *Electric–magnetic duality, monopole condensation, and confinement in* $N = 2$ *supersymmetric Yang–Mills theory*, Institute for Advanced Study, Princeton, preprint (1994).

[4] E. Witten, *Monopoles and four-manifolds*, Institute for Advanced Study, Princeton, preprint (Nov 1994).

[5] P. Kronheimer and T. Morwka, *The genus embedded surfaces in the projective plane*, (to apper, 1995).

[6] C. Montonen and D. Olive, Phys. Lett. B **72** (1977), 117.

(成 斌 译　余建明 校)