第21卷 第4期
1995年7月

自 动 化 学 报
ACTA AUTOMATICA SINICA

Vol. 21, No. 4
July, 1995

短 文

Systolic 算法和阵列结构执行固定区间平滑

慕德俊　戴冠中
(西北工业大学自动控制系　西安　710072)

摘　要

基于平方根算法提出了一种适合于并行计算的固定区间平滑的 Syatolic 算法，这种算法使得计算的快速性和数值稳定性都得到了提高. 文中还提出了一种有效的脉动 (Systloic) 阵列结构来实现此并行算法. 对容错算法、处理器的利用率及计算速度作了简要地分析.

关键词: 固定区间平滑，Systolic 算法，阵列结构.

1 引言

固定区间平滑是一种最优线性平滑，在研究卫星轨道，导弹飞行等航天技术和信号处理中都有重要的应用，但在每次迭代计算中需要复杂的矩阵运算，尤其在矩阵阶数很大，递推次数多的情况下，串行计算所需的时间很长. 超大规模集成 (VLSI) 电路的发展，使得并行处理技术得到了应用. 若对原有的串行算法开发其潜在的并行性，则可通过脉动 (Systolic) 结构加以实现. 本文基于平方根算法，Faddeev 算法，QR 分解算法提出了一种有效的适合于 Systolic 阵列实现的并行算法执行前向固定区间平滑计算，通过 $o(n^2)$ 个处理单元，使计算的复杂性由原来的 $o(n^3)$ 变为现在的 $o(n)$. 文中对容错算法、处理器的利用率及计算速度作了简要地分析.

2 前向固定区间平滑的平方根算法

设线性离散时间随机系统可表示为

$$X(k+1) = \Phi(k+1,k)X(k) + B(k)W(k), \tag{1a}$$

$$Z(k) = H(k)X(k) + V(k), \tag{1b}$$

式中 $X(k) \in R^n$，$W(k) \in R^p$，$Z(k)$，$V(k) \in R^m$ 分别为状态、量测及噪声向量，$\Phi(k+1,k) \in R^{n\times n}$ 为一步转移阵，$H(k) \in R^{m\times n}$ 的输出阵，$\{W(k)\},\{V(k)\}$ 是零均值，互不相关的白噪声序列且

本文于 1993 年 4 月 5 日收到

$$E[W(k)W^{\mathrm{T}}(j)] = Q(k)\delta_{kj},\quad E[V(k)V^{\mathrm{T}}(j)] = R(k)\delta_{kj}$$

$Q(k)$ 是对称非负定的对角阵，$R(K)$ 是对称正定对角阵.

(1) 式的前向固定区间平滑和后向信息滤波解[1]经变换可表示为

$$\hat{X}(k+1|N) = [I - G(k)K^{\mathrm{T}}(k+1)]\bar{X}(k+1|N) + G(k)[I + G^{\mathrm{T}}S(k+1)G(k)]^{-1}Y(k+1), \tag{2}$$

$$\bar{X}(k+1|N) = \Phi(k+1,k)\hat{X}(k|N), \tag{3}$$

$$\hat{X}(0|N) = [I + P(0)S(0)]^{-1}[P(0)q(0) + \hat{X}(0)],$$

$$P(k+1|N) = [I - G(k)K^{\mathrm{T}}(k+1)]\bar{P}(k+1|N)[I - G(k)K^{\mathrm{T}}(k+1)]^{\mathrm{T}} + G(k)[I + G^{\mathrm{T}}(k)S(k+1)G(k)]^{-1}G^{\mathrm{T}}(k), \tag{4}$$

$$\bar{P}(k+1|N) = \Phi(k+1,k)P(k|N)\Phi^{\mathrm{T}}(k+1,k), \tag{5}$$

$$P(0|N) = [I + P(0)S(0)]^{-1}P(0).$$

式中 $G(k)G^{\mathrm{T}}(k) = B(k)Q(k)B^{\mathrm{T}}(k)$,

$$K(k+1) = S(k+1)G(k)[I + G^{\mathrm{T}}(k)S(k+1)G(k)]^{-1},\ S(N+1) = 0,$$

$$S(k) = \Phi^{\mathrm{T}}(k+1,k)\bar{S}(k+1)\Phi(k+1,k) + H^{\mathrm{T}}(k)\bar{R}(k)H(k). \tag{6}$$

$\bar{R}(k) = R^{-1}(k) = \bar{R}^{\frac{1}{2}}(k)\cdot\bar{R}^{-\frac{\mathrm{T}}{2}}(k)$,

$$q(k) = \Phi^{\mathrm{T}}(k+1,k)\bar{q}(k+1) + H^{\mathrm{T}}(k)\bar{R}(k)Z(k), \tag{7}$$

$$Y(k) = G^{\mathrm{T}}(k)q(k), \tag{8}$$

$$\bar{S}(k+1) = [I - K(k+1)G^{\mathrm{T}}(k)]S(k+1), \tag{9}$$

$$\bar{q}(k+1) = [I - K(k+1)G^{\mathrm{T}}(k)]q(k+1),\ q(N+1) = 0. \tag{10}$$

由(6),(9)式可以看出：$S(K),\bar{S}(k+1)$ 在形式上同 kalman 滤波误差的方差阵是一致的[2],它的平方根形式的方程可表示为

$$Q_1\begin{bmatrix} S^{\frac{\mathrm{T}}{2}}(k+1)G(k) & S^{\frac{\mathrm{T}}{2}}(k+1)\Phi(k+1,k) \\ I & 0 \\ 0 & \bar{R}^{\frac{\mathrm{T}}{2}}(k)H(k) \end{bmatrix} = \begin{bmatrix} V_e^{\frac{\mathrm{T}}{2}}(k)V_e^{-\frac{1}{2}}(k)G^{\mathrm{T}}(k)S(k+1)\Phi(k+1,k) \\ 0 \quad S^{\frac{\mathrm{T}}{2}}(k) \\ 0 \end{bmatrix}. \tag{11}$$

式中 $S(k+1) = S^{1/2}(k+1)S^{\mathrm{T}/2}(k+1)$，$S^{\mathrm{T}/2}(k+1)$ 是上三角阵，Q_1 是使初始阵三角化的正交矩阵．$V_e(k) = V_e^{1/2}(k)V_e^{\mathrm{T}/2}(k) = G^{\mathrm{T}}(k)S(k+1)G(k) + I$．从(4),(5)式可以得到方差阵 $P(k+1|N)$ 的平方根形式为

$$Q_2\begin{bmatrix} P^{\frac{\mathrm{T}}{2}}(k|N)\Phi^{\mathrm{T}}(k+1,k)[I - G(k)K^{\mathrm{T}}(k+1)]^{\mathrm{T}} \\ [I + G^{\mathrm{T}}(k)S(k+1)G(k)]^{-\frac{1}{2}}G^{\mathrm{T}}(K) \end{bmatrix} = \begin{bmatrix} P^{\frac{\mathrm{T}}{2}}(k+1|N) \\ 0 \end{bmatrix}. \tag{12}$$

式中 $P(k|N) = P^{1/2}(k|N)P^{\mathrm{T}/2}(k|N)$，$P^{\mathrm{T}/2}(k|N)$ 为上三角阵．Q_2 是正交矩阵．

3 Systloic 阵列结构实现前向固定区间平滑计算

平方根算法具有较好的数值稳定性，并能很容易地映射到阵列结构上．为了使数据快速平稳地流动，本文使用两个阵列(一个梯形阵列和一个三角阵列）组合的方式共同完成此算法，以及采用了处理单元在不同阶段完成不同的运算功能的方法．整个系统的结构简图见图 1，其中，阵列 1 是由左边大小为 $p \times p$ 的三角阵列与 $p \times n$ 的矩形阵列组成的梯形阵列，阵列 2 是 $n \times n$ 的三角阵列加上上部 n 个单元的一维阵列组成，整个计算过程可分为四步进行．

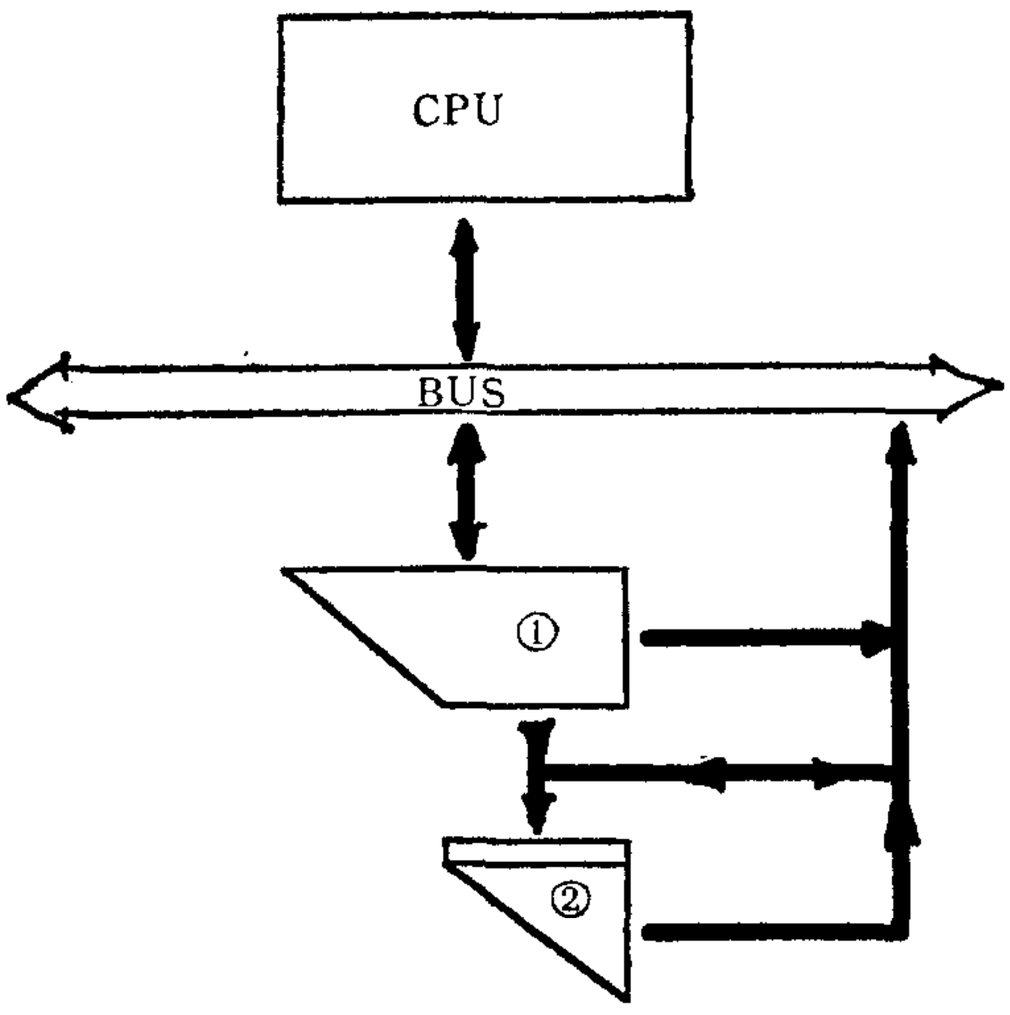

图 1 固定区间平滑的阵列结构简图

第一步．计算后向递推信息平方根阵 $S^{T/2}(k)$

这一步的计算是通过阵列 1 和 2 组成的三角阵列对(11)式的初始矩阵进行 QR 分解来完成，由于 I 是已知阵且 $\bar{R}^{\frac{T}{2}}(k) \times H(k)$ 也很容易预先得到，因此，可将这两个矩阵预先存入阵列 1 左边的三角阵列和阵列 2 中(见图 2(a))，$S^{T/2}(k)G^{T}(k)$ 和 $S^{T/2} \times (k+1)\Phi(k+1,k)$ 输入阵列后得到 $V_e^{T/2}(k)$、$V_e^{-1/2}(k)G^{T}(k)S(k+1)\Phi(k+1,k)$和 $S^{T/2}(k)$，分别存入阵列 1 的左右两部分及阵列 2 中(见图 2(b))．

第二步．计算 $q(K)$ 和矩阵相乘计算

为了计算 $q(K)$，可对(7),(10)式进行 Faddeev 运算，(7),(10)式整理后得

$$q^{T}(k) = [q^{T}(k+1)\Phi(k+1,k) + Z^{T}(k)\bar{R}(k)H(k)]$$

$$- [q^{T}(k+1)G(k)][V_e(k)]^{-\frac{T}{2}}[V_e^{-\frac{1}{2}}G^{T}(k)S(k+1)\Phi(k+1,k)].$$

由于等式右边最后一项及 $V_e^{T/2}(k)$ 阵已在阵列 1 中，因此仅需将等式右边的前两个向量输入阵列 1，产生 $q^{T}(k)$ 存入阵列 2 上面的一维阵列中．为了产生下次迭代所需的初始矩阵，需将 $\Phi^{T}(k,k-1)$，$G^{T}(k-1)$ 输入阵列 2 分别与里面的 $q^{T}(k)\Phi(k,k-1)$、$S^{T/2}(k)$ 相乘，得到 $q^{T}(k)\Phi(k,k-1)$，$q^{T}(k)G(k-1)$，$S^{T/2}(k)\Phi(k,k-1)$，$S^{T/2}(k) \times G(k-1)$ 并输出，再将 $q^{T}(k)\Phi(k,k-1)$ 与 $Z^{T}(k)\bar{S}(k)H(k)$ 相加(见图 2(b))．

第三步．计算 $\Phi^{T}(k+1,k)[I - G(k)K^{T}(k+1)]^{T}$，$G(k)[I + G^{T}(k)S(k+1) \times G(k)]^{-1}Y(k+1)$ 及 $[I + G^{T}(k)S(k+1)G(k)]^{-1/2}G^{T}(k)$

这一步是计算(2),(12)式所需的初始矩阵，输入 $G(k)$ 和 $\Phi(k+1,k)$ 到阵列 1(见图 2(c)) 执行 Gaussian 消去运算，在阵列下部和右边输出 $[I - G(k)K^{T}(k+1)]\Phi(k+1,k)$，$[I + G^{T}(k)S(k+1)G(k)]^{-1/2}G^{T}(k)$ 并存贮以便前向平滑计算．输入 $q^{T}(k+1)G(k+1)$，与三角阵列里的矩阵进行逆乘运算[3]，得到的 $[I+G^{T}(k)S(k+1)G(k)]^{-1/2} \times$

$Y(k+1)$ 再与右边矩形阵列里的 $[I+G^T(k)S(k+1)G(k)]^{-1/2}G^T(k)$ 相乘，得到 $G(k)[I+G^T(k)S(k+1)G(k)]^{-1}Y(k+1)$ 从阵列下方输出.

第四步. 前向平滑计算

输入 $\Phi^T(k+1,k)[I-G(k)K^T(k+1)]^T$ 到阵列 2 (图 2(d)) 与其中的 $P^{T/2}(k|N)$ 相乘,便可得到 $P^{T/2}(k|N)\Phi^T(k+1,k)[I-G(k)K^T(k+1)]^T$，再与 $[I+G^T(k)S(k+1)G(k)]^{-1/2}G^T(k)$ 一起输入阵列 2 完成(12)式的 QR 分解，得到 $P^{T/2}(k+1|N)$ 存在

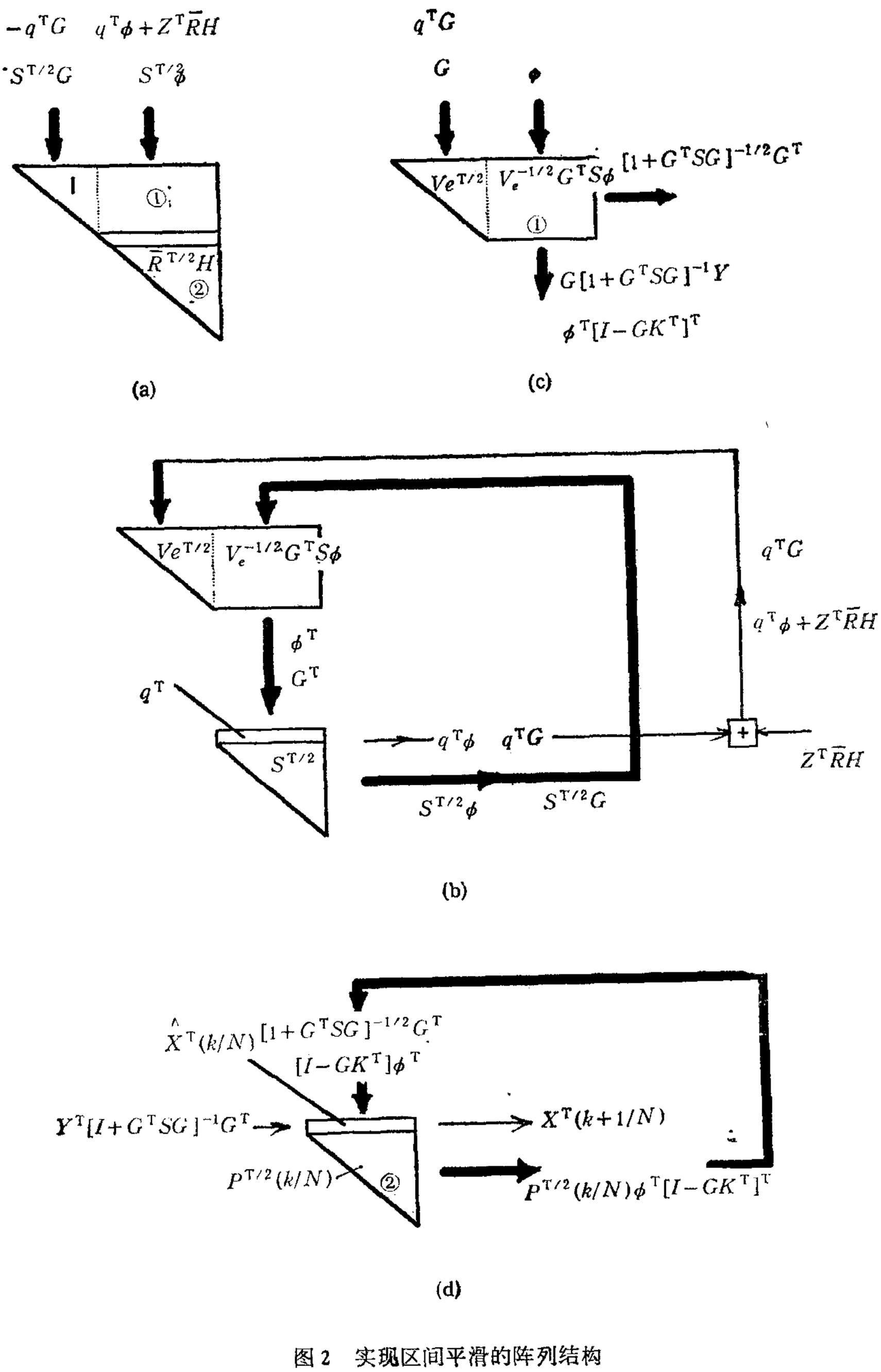

图 2　实现区间平滑的阵列结构

阵列里;与此同时,将一维阵列中的 $\hat{X}^{\mathrm{T}}(k|N)$ 与输入的 $\Phi^{\mathrm{T}}(k+1,k)[I-G(k)K^{\mathrm{T}}(k+1)]^{\mathrm{T}}$ 和 $Y^{\mathrm{T}}(k+1)[I+G^{\mathrm{T}}(k)S(k+1)G(k)]^{-1}G^{\mathrm{T}}(k)$ 进行乘加运算得到 $\hat{X}(k+1|N)$.

4 计算周期、处理器利用率及容错算法

阵列 1 和 2 共有 $n+(n+p)(n+p+1)/2$ 个处理单元. 由时序分析可知, 在有效地组织数据流进程的情况下, 后向信息滤波的迭代周期为 $2n+2p$ 个单位时间; 从图 2(d)可以看出: 前向平滑的迭代周期为 $2n+p$ 个单位时间,因此, 对一个点的滤波和平滑共需 $4n+3p$ 个单位时间.

阵列 1 共有 $p(2n+p+1)/2$ 个处理单元,每个单元在每次递推中共进行了 $2n+2$ 个单位时间的计算,阵列 2 共有 $(n+3)n/2$ 个处理单元, 在每次递推中,上面的一维阵列和三角阵列里的每个单元各进行了 $2n+p$ 和 $2n+2p$ 个单位时间的计算,因此, 整个阵列的利用率为: $2[p(2n+p+1)(n+1)+(2n+p)n+(n+1)n(n+p)]/\{[n(n+3)+p(2n+p+1)](4n+3p)\}$, 结果约得 45% 左右.

一种简单而有效的容错方案是检验和 (Checksum) 与加权检验和 (Weighted checksum) 方法[4]. 由于本文所提出的并行算法使得阵列单元在不同的阶段执行不同的运算,因此按照每个阶段单元运算的特性,在不同的运算阶段需要不同的容错方案, 本文使用了四种矩阵计算, QR 分解时,容错算法采用了行检验和 (RCS) 方法[5], Gaussian 消去计算的容错算法也采用了行检验和法,矩阵与矩阵相乘的容错算法则采用全检验法,矩阵逆乘的容错算法采用列 (CCS) 检验法. 以上容错算法只能用于错误检测, 错误定位和修正算法有待于进一步研究.

5 结论

本文采用的 Systolic 算法并行计算固定区间平滑,既可提高计算的数值稳性也可使计算的速度大大提高, 这种算法很容易映射到阵列结构上. 处理器单元在不同阶段执行不同运算既可减少硬件的需求,又可使结构紧凑,处理器的利用率又可提高,因此,本文所提出的并行算法和阵列结构是一种有效的执行固定区间平滑的方法. 另外本文所提出的容错算法也是一种有效的错误探测方案,错误定位和修正算法需进一步研究.

参 考 文 献

[1] Watanabe K. Adaptive estimation and control, Prentice Hall, 1992.

[2] Irwin G W. A Systolic architecture for square-root covariance Kalman filter Systolic Array Processor. Prentice Hall, 1989.

[3] Mcwhirter J G. Alorithmic engineering in adaptive signal processing. IEE Pt F, 1992, **139**: 226—232.

[4] Huang K H. Algorithm-based fault tolerance for matrix operations, *IEEE Trans. Computer*. 1984, **C-33**: 518—529.

[5] Chen C Y. Tault-tolerance systems for the computation of eigenvalue and singular values. *SPIE*, 1986, **696**: 228—237.

A SYSTOLIC ALGORITHM AND ARRAY FOR FIXED-INTERAAL SMOOTHER

MU DEJUN DAI GUANZHONG

(*Department of Automatic Control, Northwestern Polytechnical University Xi'an* 710072)

ABSTRACT

A systolic algorithm for parallel forward-pass fixed-interval smoother is derived in this paper, efficient systolic arrays for performing the algorithm are also developed. The properties of real-time and numerical stability of the parallel algorithm are improved, and the time-steps for a complete iteration and the utilization for the processors are analysized. Efficient algorithms based on fault-toleration scheme are also proposed.

Key words: Forward-pass fixed-interval smoother, systolic algorithm, systolic array.