中山大学学报（自然科学版）
第30卷 第3期 ACTA SCIENTIARUM NATURALIUM Vol.30 №.3
1991年 UNIVERSITATIS SUNYATSENI 1991

自适应格形算法的并行实现

胡梦佑* 陈钧量
（无线电电子学系）

摘 要

本文提出一种基于自适应格形算法的并行实现方法。它节省硬件，处理速度高，但不影响算法的性能，适用于高速、高性能的实时处理。

关键词 自适应格形算法，阶迭归状态方程，阶迭归输出方程

自适应信号处理在通信、图像处理、语音处理和生物医电处理等领域得到广泛应用[1～3]。但自适应信号处理算法本身还存在着两方面的问题：一是算法的数值稳定性[4～6]，即由有限精度运算引起的算法不适定性，这将另文讨论。另一是算法的速度。近年来，已出现不少快速卡尔曼算法[3]和快速分块算法[4]，但仍都存在着数值稳定性问题。自适应格形算法在数值特性方面是优于快速卡尔曼算法和分块算法，但运算量较大。若将格形算法按顺序实现，则难以满足实时处理要求。文献[7]提出一种阵列处理（Array Processing）方法，但存在硬件利用率低的缺陷。文献[8]给出了三种流水线式多处理器结构，它们运算速度高，但硬件实现复杂，尤其是第三种结构。本文提出一种适用于格形算法的并行实现方法。

1 格形算法

目前所出现的多种格形算法，其基本构架完全相同。这里仅列出预加窗的非归一化算法[2]作为讨论的例子。

输入参数：p是格形最大阶数；W是指数加权因子或遗忘因子（通常取值0.95～0.998）；$X(T)$是T时刻的数据采样输入。

变量定义：$k(T,n)$是部份自相关系数；$R(T,n)$是Likelihood变量；$e_f(T,n)/e_b(T,n)$是前向/后向预测误差；$p_f(T,n)/p_b(T,n)$是前向/后向预测误差方差；$k_f(T,n)/k_b(T,n)$是前向/后向反射系数。

初始化：$e_b(0,n)=k(0,n)=0$；$R(0,n)=1$

$p_f(0,n)=p_b(0,n)=\delta$ （δ为一小正数）

FOR T = 1 TO final DO

$R(T,0)=1$；$e_f(T,0)=e_b(T,0)=X(T)$

本文1990年5月23日收到

* 1988级硕士研究生

$$p_f(T,0)=p_b(T,0)=W\cdot p_f(T-1,0)+X(T)\cdot X(T)$$

FOR n = 0 TO p − 1 DO

$$k(T,n+1)=k(T-1,n+1)\cdot W+e_f(T,n)\cdot e_b(T-1,n)/(1-R(T-1,n)) \quad (1)$$

$$k_f(T,n+1)=k(T,n+1)/p_f(T,n) \quad (2)$$

$$k_b(T,n+1)=k(T,n+1)/p_b(T-1,n) \quad (3)$$

$$e_f(T,n+1)=e_f(T,n)-k_b(T,n+1)\cdot e_b(T-1,n) \quad (4)$$

$$e_b(T,n+1)=e_b(T-1,n)-k_f(T,n+1)\cdot e_f(T,n) \quad (5)$$

$$p_f(T,n+1)=p_f(T,n)-k_b(T,n+1)\cdot k(T,n+1) \quad (6)$$

$$p_b(T,n+1)=p_b(T-1,n)-k_f(T,n+1)\cdot k(T,n+1) \quad (7)$$

$$R(T,n+1)=R(T,n)+e_b(T,n)\cdot e_b(T,n)/p_b(T,n) \quad (8)$$

2 并行实现

从式(4)～(8)可以看出，对每一阶的迭归，各式的计算依赖低一阶的迭代输出和式(1)～(3)的计算结果；而式(1)～(3)的计算仅依赖低一阶的迭归输出和采样输入（对零阶而言）．因此，我们可以将一次阶迭代归纳为两类计算，分别表示如下

$$\vec{K}(T,n+1)=\vec{A}(T)\cdot\vec{K}(T-1,n+1)+\vec{B}(T) \quad (9)$$

$$\vec{F}(T,n+1)=\vec{C}(T)\cdot\vec{F}(T,n)+\vec{D}(T) \quad (10)$$

其中

$$\vec{K}(T,n+1)=[k(T,n+1),\ k_f(T,n+1),\ k_b(T,n+1)]'$$

$$\vec{F}(T,n+1)=[e_f(T,n+1),\ e_b(T,n+1),\ p_f(T,n+1),\ p_b(T,n+1),\ R(T,n+1)]'$$

$$\vec{A}(T)=W\times\mathrm{diag}[1,\ p_f(T-1,n)/p_f(T,n),\ p_b(T-2,n)/p_b(T-1,n)]$$

$$\vec{B}(T)=M[1,\ 1/p_f(T,n),\ 1/p_b(T-1,n)]'$$

$$M=e_f(T,n)\cdot e_b(T-1,n)/(1-R(T-1,n))$$

$$\vec{C}(T)=\mathrm{diag}[1,\ Z^{-1},\ 1,\ Z^{-1},\ 1]$$

$$\vec{D}(T)=[-k_b(T,n+1)\cdot e_b(T-1,n),\ -k_f(T,n+1)\cdot e_f(T,n),\ -k_b(T,n+1)\cdot k(T,n+1),\ -k_f(T,n+1)\cdot k(T,n+1),\ e_b(T,n)\cdot e_b(T,n)/p_b(T,n)]'$$

式中上标“′”表示转置，Z^{-1}为单位延迟算子．

式(9)称为阶迭归状态方程，简称状态方程；式(10)称为阶迭归输出方程，简称输出方程．式(9)和(10)的流图见图1．

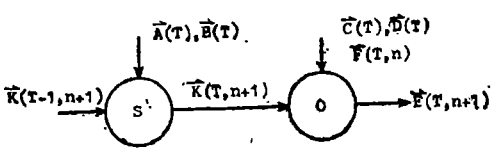

图1 状态块（S）和输出块（O）的连结

Fig.1 Connection between state block and output block

这里，对于不同类型的格形算法，矩阵$\vec{A}$，$\vec{B}$，$\vec{C}$和$\vec{D}$的基本元素是不同的．但$\vec{A}$和

$\vec{B}$的计算仅依赖于低一阶的迭代输出，而$\vec{C}$和$\vec{D}$的计算依赖于低一阶的迭归输出和当前迭代状态的输出。

同时考虑p个采样输入时，可分别得到p个状态方程和p个输出方程

$$\begin{aligned}
&\vec{K}(T,n+1)=\vec{A}(T)\cdot\vec{K}(T-1,\ n+1)+\vec{B}(T)\\
&\vec{K}(T+1,n+1)=\vec{A}(T+1)\cdot\vec{K}(T,n+1)+\vec{B}(T+1)\\
&\qquad\cdots\qquad\qquad\cdots\qquad\qquad(11)\\
&\vec{K}(T+p-1,\ n+1)=\vec{A}(T+p-1)\cdot\vec{K}(T+p-2,\ n+1)+\vec{B}(T+p-1)
\end{aligned}$$

和

$$\begin{aligned}
&\vec{F}(T,n+1)=\vec{C}(T)\cdot\vec{F}(T,n)+\vec{D}(T)\\
&\vec{F}(T+1,\ n+1)=\vec{C}(T+1)\cdot\vec{F}(T+1,\ n)+\vec{D}(T+1)\\
&\qquad\cdots\qquad\qquad\cdots\qquad\qquad(12)\\
&\vec{F}(T+p-1,\ n+1)=\vec{C}(T+p-1)\cdot\vec{F}(T+p-1,\ n)+\vec{D}(T+p-1)
\end{aligned}$$

由此可以得到p个输入时单阶的迭代流图见图 2。图中状态块S_1—S_p完成式(11)运算，输出块O_1—O_p完成式(12)运算。

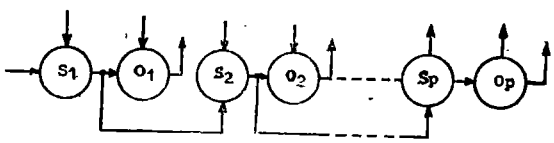

图 2　基本处理单元

Fig.2 Basic processing unit

同理，不难给出p阶p个采样输入时的流图见图 3。图中$S_{i,j}$和$O_{i,j}$分别表示第i个输入、第j阶的状态块和输出块。

从图 3 看出，同一阶的运算结构完全相同，对不同采样输入的运算结构也完全相同。这样，我们可以得到图 4 所示的并行处理系统。其中，输入单元由一个A/D变换器和p个寄存器组成，完成将数据由串行到并行的转换，并暂存这些数据。基本处理单元由p个单片机构成，在控制单元的作用下，完成同步并行运算。控制单元由单片机担任。输出单元由一个D/A变换器和p个寄存器构成，完成将数据由并行到串行的转换，并暂存p个数据。本系统是一个同步并行系统，采用单时钟进行同步，其同步脉冲由控制单元提供。整个系统的工作过程用流程表示见图 5。

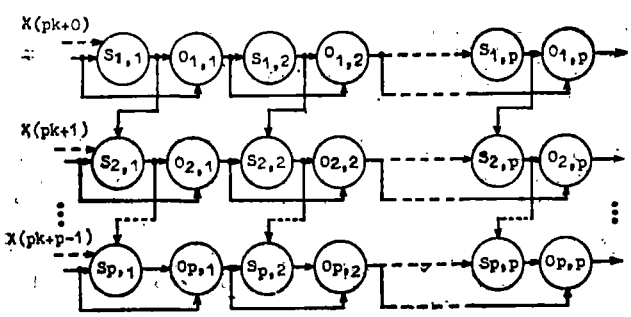

图 3　p阶p个采样输入时状态块与输出块的连结

Fig. 3 Connection between state blocks and output blocks for p orders with p input samples

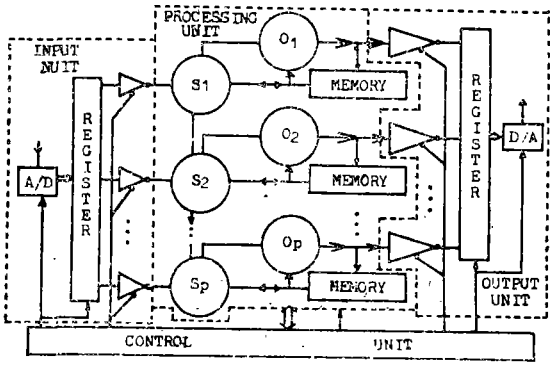

图4 并行处理系统

Fig.4 A parallel system

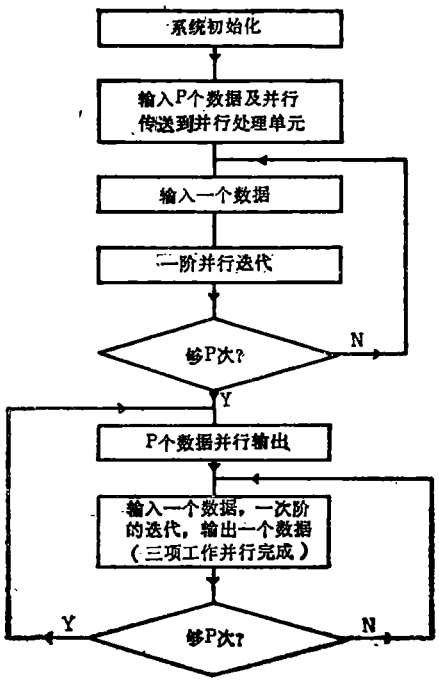

图5 系统工作流程

Fig.5 Flowchart of system operation

整个系统的初始化工作分为两步，首先将输入单元初始化，再于输入单元接收前p个采样的同时，对系统的其余部份初始化。阶迭归的初始化可视为最低一阶的输入。因此，可在系统初始化的同时来完成。

若本系统用于辨识等方面的应用，则额外的计算量由控制单元完成。

参 考 文 献

1 Jou I C, Hu Y H, Feng W S. Proc of IEEE ISCAS'86, 1986:235
2 Honig M L, Messerchmitt D G. Adaptive Filters: Structures, Algorithms and Applications. Boston, MA, Kluwer Academic, 1984
3 Alexander S T. Adaptive Signal Processing. Springer-Verlag, NY. 1986
4 Yu X H, He Z Y. IEEE Trans ASSP, 1988; 36: 392
5 Lin D W. IEEE Trans ASSP, 1984; 32: 998
6 Ardalan S H, Faber L J. IEEE Trans ASSP, 1988;36: 349
7 Meng T H Y, Messerchmitt D G. IEEE Trans ASSP, 1987;35:455
8 Meyer M D, Agrawal D P. Proc of IEEE ICASSP'88, 1988:1580

A Parallel Implementation of Adaptive Lattice Algorithms

Hu Mengyou Chen Junliang*

Abstract

A parallel implementation based on adaptive lattice algorithms is proposed. The implementation is hardware saving and high operational speed but it has no effect on the algorithm performance, which improves the performance for high speed real-time signal processing.

Keywords adaptive lattice algorithm, order-recursive state equation, order-recursive output equation

• Department of Radio and Electronics