یک معماری سخت‌افزاری کارآمد برای کنترلرهای پیش‌بین مدل

محمد حسین منتظری[۱]، محسن طاهری[۲]، مهدی سعیدی[۳]، حسن طاهری[۴]، مرتضی صاحب الزمانی[۵]

چکیده

کنترلرهای پیش‌بین مدل (MPC) اخیراً در سیستم‌های کنترلی گوناگونی استفاده شده‌اند. این در حالی است که حجم فوق‌العاده بالای محاسبات موردنیاز، باعث کندی آنها شده و از این‌رو، کاربرد این کنترلرها به سیستم‌های کند محدود شده است. این مقاله در حقیقت تلاشی در جهت استفاده از کنترلرهای MPC در کاربردهای سریع با استفاده از پیاده‌سازی سخت‌افزاری است. در این مقاله یک الگوریتم شبه‌بهینه برای استفاده از کنترلرهای MPC در شبکه‌های کامپیوتری پیشنهاد شده است. بعلاوه به منظور افزایش سرعت پردازش، یک معماری کارآمد برای پیاده‌سازی سخت‌افزاری آنها ارائه شده است. نتایج آزمایشات نشان می‌دهد که با استفاده از معماری ارائه شده می‌توان از کنترلرهای MPC در کاربردهای با سرعتهای بالا نظیر کنترل زمانبندی صفها در شبکه‌های کامپیوتری استفاده نمود.

کلمات کلیدی

کنترلرهای پیش‌بین مدل و برنامه‌ریزی مربعی.

An Efficient Hardware Architecture for Model Predictive Controllers

Mohammad H. Montazeri, Mohsen Taheri, Mehdi Saeedi, Hassan Taheri, Morteza Saheb Zamani
Amirkabir University of Technology

Abstract

Model predictive controllers (MPC) have been used in various control systems, recently. However, the enormous processes required in these controllers make them relatively slow and their main application is limited to slow control systems. This paper is an attempt to use MPC controllers in high speed applications by using efficient hardware architectures. In this paper, we propose a semi-optimal algorithm to use MPC controllers in computer network systems. Furthermore, an efficient hardware implementation for these controllers has been presented to increase their processing speed. The experimental results show that by using the proposed architecture, we can use MPC controllers in high speed applications such as controlling queue schedulers in computer networks very efficiently.

Keywords

Model Predictive Controllers, Quadratic Programming, Queue Scheduling.

[1] دانشجوی کارشناسی ارشد، دانشکدهٔ مهندسی برق، دانشگاه صنعتی امیرکبیر، پست الکترونیک: mhmontazerisa@aut.ac.ir
[2] دانشجوی کارشناسی ارشد، دانشکدهٔ مهندسی برق، دانشگاه صنعتی امیرکبیر، پست الکترونیک: mohsentaheri@aut.ac.ir
[3] دانشجوی دکترا، دانشکدهٔ مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، پست الکترونیک: saeedi@ce.aut.ac.ir
[4] دکترا، عضو هیأت علمی، دانشکدهٔ مهندسی برق، دانشگاه صنعتی امیرکبیر، پست الکترونیک: htaheri@aut.ac.ir
[5] دکترا، عضو هیأت علمی، دانشکدهٔ مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، پست الکترونیک: szamani@ce.aut.ac.ir

۱- مقدمه

امروزه نیاز به کنترلرهای سریع در صنعت برای کنترل فرآیندهای فوق‌العاده حساس بسیار ضروری شده است و این درحالی است که استفاده از کنترلرهای معمول PID[1] برای این امر با مشکلاتی همراه است. کنترلرهای پیش‌بین مدل[2] (MPC) تاکنون تنها در کاربردهای با سرعت بسیار پایین نظیر کنترل فرایندهای شیمیایی که نیاز به نرخ کنترل بسیار پایینی دارند، مورد استفاده قرار گرفته است و از این‌رو حجم فوق‌العاده بالای محاسبات موردنیاز برای آنها، به عنوان یک مشکل در استفاده به شمار نمی‌آمده است. در هر صورت امروزه نیاز به سرعت بالا در کنترل فرآیندهای سریع نظیر کنترل زمانبندی صف‌های مسیریابهای شبکه‌های کامپیوتری، کاربردهای جدیدی را برای کنترلرهای MPC بوجود آورده است و از این‌رو مشکل کندی و نیز حجم محاسبات زمان‌گیر این کنترلرها باید برای استفاده حل شود.

یکی از روش‌های کارآمد برای بهبود زمان اجرای کنترلرهای پیش‌بین مدل، استفاده از پیاده‌سازی سخت‌افزاری آنها می‌باشد. با توجه به اینکه تقریباً تمامی پیاده‌سازی‌های موجود برای کنترلرهای MPC به صورت نرم‌افزاری هستند و بعلاوه چندین پیاده‌سازی سخت‌افزاری ارائه شده نیز دارای معماری ناکارآمدی می‌باشند، در این مقاله به ارائهٔ یک معماری کارامد برای پیاده‌سازی سخت‌افزاری کنترلرهای MPC پرداخته شده است.

ادامهٔ این مقاله به این صورت سازماندهی شده است: ابتدا در بخش ۲ به معرفی کنترل‌کننده‌های پیش‌بین مدل پرداخته و بحث‌های تئوری موردنیاز برای استفاده در ادامهٔ مقاله ارائه می‌شود. سپس در بخش ۳ به بررسی پیشینهٔ کار پرداخته شده است. در بخش ۴ الگوریتم RLS برای یافتن یک مدل خطی متغیر با زمان از سیستم تحت کنترل ارائه و معادلات مورد استفاده تشریح خواهد شد. در ادامه و در بخش ۵ یک سیستم تحت کنترل نمونه بررسی و نحوهٔ اعمال پیش‌بین مدل به آن تشریح می‌شود. سپس در بخش ۶، به معرفی یک معماری برای پیاده‌سازی سخت‌افزاری کنترل‌کنندهٔ پیش‌بین مدل پرداخته و در بخش ۷ نتایج آزمایشات ارائه خواهد شد و در نهایت این مقاله با نتیجه‌گیری و ارائهٔ پیشنهاد برای کارهای آتی در بخش ۸ خاتمه می‌یابد.

۲- کنترلرهای پیش‌بین مدل

در کنترلرهای PID روش کنترل کردن با انتظار برای رخداد خطا و سپس اقدام جهت رفع آن می‌باشد و از این‌رو به این کنترلرها کنترلرهای حلقه‌بسته[3] گفته می‌شود. این مسأله باعث می‌گردد که در کاربردهای حیاتی و پرمخاطره نظیر کنترل بیماری‌ها نتوان از کنترلرهای PID استفاده نمود. به عبارت دیگر باید گفت که در برخی از کاربردها نظیر آنچه ذکر شد باید از کنترلری بهره گرفت که قابلیت پیش‌بینی و جریمه خطا را قبل از وقوع داشته باشد و به این ترتیب کنترلرهای پیش‌بین مدل معرفی شدند.

بعلاوه از آنجایی‌که اکثر سیستم‌های موجود به صورت غیرخطی عمل می‌کنند، استفاده از روش‌های پیش‌بینی خطی برای آنها مناسب نیست و به همین خاطر استفاده از قابلیت بهینه‌سازی غیرخطی (مربعی) موجود در کنترلرهای پیش‌بین مدل برای آنها بسیار مناسب است و این امر بر محبوبیت این کنترلرها می‌افزاید.

در MPC ها از دو مفهوم "افق دید" و "افق کنترل" استفاده می‌شود. افق دید بیانگر طول پیش‌بینی آینده در هر گام می‌باشد و افق کنترل نیز به تعداد گامهایی از آینده که ورودیهای بهینهٔ سیستم برای آنها محاسبه می‌شوند، اطلاق می‌گردد. بدیهی است طول افق کنترل باید از طول افق دید کمتر و یا مساوی با آن باشد چراکه استفاده از گامهایی که در آنها محاسبه‌ای صورت نگرفته است، باعث تزریق خطا خواهد شد.

در حالت ساده می‌توان یک سیستم را با معادلات فضای حالت به صورت زیر مدل کرد:

$$\begin{cases} x(n+1) = Ax(n) + Bu(n) \\ y(k) = C_y x(k) \\ z(k) = C_x x(k) \end{cases} \quad (1)$$

که در آن x یک بردار حالت m تایی و u بردار ورودی سیستم می‌باشد. بعلاوه y نیز یک بردار m تایی از خروجی‌هایی است که باید اندازه‌گیری شوند و z نیز یک بردار m تایی از خروجی‌هایی است که باید کنترل شوند. ماتریسهای A، B، C_y و C_x نیز ماتریس ضرایب می‌باشند. با توجه به اینکه متغیرهای y و z معمولاً یکسان هستند و یا اشتراکات زیادی با هم دارند، در این مقاله بدون از دست دادن کلیت مسأله فرض می‌شود که y و z یک بردار می‌باشند. بر این اساس، در کنترلرهای پیش‌بین از الگوریتم زیر استفاده خواهد شد:

- اندازه‌گیری خروجی مطلوب سیستم
- پیش‌بینی آینده سیستم با استفاده از مدل
- حل یک معادلهٔ بهینه‌سازی به نحوی که ورودی‌های u(k) یک تابع هدف را کمینه کنند.
- اعمال ورودی‌های u(k) به سیستم
- یک گام به جلو رفتن و بازگشت به مرحلهٔ اول

در حالت کلی می‌توان گفت که مهمترین وظیفهٔ یک کنترلر پیش‌بین مدل، بهینه‌سازی یک تابع هزینهٔ غیرخطی (مربعی) است که به صورت زیر نشان داده می‌شود:

$$J_P(k) = \sum_{i=H_w}^{H_p} \left\| y(n+i) - s(n+i) \right\|^2 w(i) + \sum_{i=0}^{H_u} \left\| \Delta u(n+i) \right\|^2 R(i) \quad (2)$$

در این معادله، s خط سیر مرجع یا مجموعهٔ خروجیهای مطلوب، Δu میزان تغییر ورودی سیستم در هر گام کنترلی i، H_p طول افق دید و H_u طول افق کنترل است. ماتریسهای W و R نیز به ترتیب برای وزن‌دهی به خروجیهای سیستم برای دخالت در محاسبهٔ تابع هدف و همچنین برای وزن‌دهی به گامهای مختلف برای اعمال کنترل مورد استفاده قرار می‌گیرند. H_w نیز به منظور شروع فرایند کنترلی از چند گام جلوتر از صفر استفاده شده است تا کنترلر فرصت کافی برای شناخت سیستم داده باشد و پس از آن شروع به جریمهٔ خطا نماید.

پیش‌بینی، وظیفهٔ مهم دیگری است که باید توسط کنترلر پیش‌بین مدل صورت پذیرد. این‌کار توسط استفاده از مدل فضای حالت بیان شده به صورت زیر انجام می‌گیرد:

$$\begin{aligned} &y(n+1) = Ay(n) + Bu(n) \\ &y(n+2) = A^2 y(n) + ABu(n) + Bu(n+1) \\ &\vdots \\ &y(n+H_P) = A^{H_P} y(n) + A^{H_P-1}Bu(n) + \\ &\qquad \cdots + Bu(n+H_P-1) \end{aligned} \tag{3}$$

اگر در معادلات فوق مقادیر u(n+i) را با عبارات زیر جایگزین کنیم:

$$\begin{aligned} &u(n) = \Delta u(n) + u(n-1) \\ &u(n+1) = \Delta u(n+1) + \Delta u(n) + u(n-1) \\ &\vdots \\ &u(n+H_u-1) = \Delta u(n+H_u-1) + \\ &\qquad \cdots + \Delta u(n) + u(n-1) \end{aligned} \tag{4}$$

می‌توان به معادله به شکل ماتریسی زیر رسید:

$$Y(n) = \Psi y(n) + \Gamma u(n-1) + \Theta \Delta U(n) \tag{5}$$

که در آن دو جملهٔ اول گذشته و جملهٔ سوم آینده را مشخص کرده است. در این قسمت ε را به صورت زیر تعریف می‌کنیم:

$$\varepsilon(n) = \tau(n) - \Psi y(n) - \Gamma u(n-1) \tag{6}$$

که در آن

$$\tau(n) = [r(n+H_W)\cdots r(n+H_P)]^T \tag{7}$$

است. با جایگزینی عبارات فوق در معادلهٔ برنامه‌ریزی مربعی[۴] خواهیم داشت:

$$\begin{aligned} J &= \| \Theta \Delta U(n) - \varepsilon(n) \|_W^2 + \| \Delta U(n) \|_R^2 \\ &= const - \Delta U(n)^T G + \Delta U(n)^T H \Delta U(n) \end{aligned} \tag{8}$$

که در آن:

$$\begin{aligned} G &= 2\Theta^T W \varepsilon(n) \\ H &= \Theta^T W \Theta + R \end{aligned} \tag{9}$$

به این ترتیب اگر مسأله دارای محدودیت نباشد ورودی بهینه با استفاده از مشتق‌گیری و برابر صفر قرار دادن جواب بدست می‌آید. در غیر اینصورت، می‌توان شکل استاندارد زیر را برای برنامه‌ریزی مربعی در نظر گرفت:

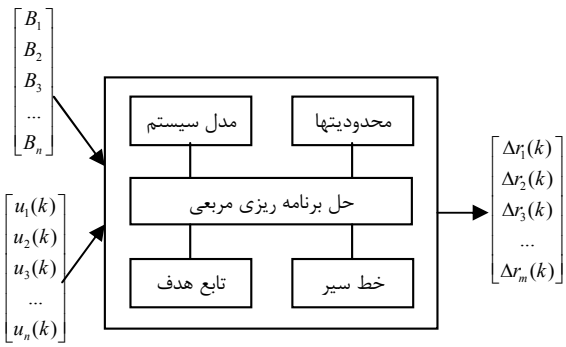
$$\min f(u) = \frac{1}{2} u^T G u + g^r u + \sum_i d_i(u) \tag{10}$$

مشکل این معادله می‌تواند با حل عددی تقریبی f(u) توسط یک تابع برنامه‌ریزی مربعی حول u، با بدست آوردن گرادیان $\nabla f(u)$ و H(u)(Hessian) و با استفاده از تکرار، انجام شود که در واقع سهم عمده پیچیدگی این الگوریتم به محاسبه ماتریس H^{-1} مربوط می‌شود.

$$u^{(k+1)} = u^{(k)} - H^{-1}(u^{(k)}).\nabla f(u^{(k)}) \tag{11}$$

ساختار اصلی کنترلر پیش بین مدل در شکل ۱ نشان داده شده است. در این کنترلرها، از یک مدل که با کمک مقادیر گذشته و حال به پیش‌بینی خروجیهای آیندهٔ فرآیند تحت کنترل می‌پردازد، استفاده می‌شود. همان طوری که در شکل ۱ نیز دیده می‌شود کنترلر پیش‌بین مدل با استفاده از روش حل کمترین مربعات خطا و با استفاده از ورودیها و محدودیتهای لازم و البته با بکارگیری مدل سیستم و یک خط سیر مرجع تغییرات لازم در مرحلهٔ بعد از کار سیستم را نسبت به حالت قبلی به دست آورده و به سیستم اعمال می‌نماید.

در کنترل پیش‌بین مدل از مدلهای متفاوتی می‌تواند استفاده شود. یکی از مهمترین این مدلها، مدل پاسخ پله می‌باشد که می‌تواند بوسیلهٔ اندازه‌گیری خروجی سیستم وقتی با یک تابع پله تحریک شده است به دست آید. یک مدل مشابه، مدل پاسخ ضربه است. مدلهای پاسخ پله و پاسخ ضربه مدلهای استاتیک هستند. از آنجا که در اکثر کاربردها به مدل پویایی نیاز می‌باشد، در این مقاله از مدل فضای حالت و با بکارگیری الگوریتم RLS^{Δ} که یک الگوریتم مدل تطبیقی است، استفاده می‌شود. خصوصیت ویژهٔ مدل فضای حالت، توصیف فرآیندهای چند متغیره است که در بسیاری از کاربردها بسیار مهم می‌باشد.



شکل ۱- ساختار کلی کنترلر پیش‌بین مدل

۳- پیشینهٔ کار

اولین کنترلرهای پیش‌بین مدل در دههٔ هفتاد میلادی و در صنعت بوجود آمدند و از این‌رو مبدع اولیه برای آنها معرفی نشده است. اما به هرحال این کنترلرها نخستین بار توسط Richalet و همکارانش در [1] مطرح شدند. همان طوری که ذکر شد بهینه‌سازی تابع هدف مهمترین بخش یک کنترلر پیش‌بین مدل می‌باشد که بسیار زمان‌گیر است و از این‌رو عمدهٔ کارها بر این بخش تمرکز یافته است که در این مقاله هیچ توجهی به آن نشده بود.

در اکثر مقالاتی که تاکنون برای افزایش سرعت این کنترلرها منتشر شده است، با استفاده از تقریب برای ساده‌سازی و یا راه‌حلهای نرم‌افزاری به بهبود کارایی می‌پردازند. به عنوان مثالی در این خصوص می‌توان به دو مقالهٔ [2] و [3] اشاره کرد که در آنها سعی می‌شود که توسط ساده‌سازیهای ریاضی، محاسبات مربوط به حل برنامه‌ریزی مربعی کاهش یابد.

در مقالات دیگر نظیر [4] و نیز در چندین کار در قالب رسالهٔ کارشناسی ارشد و دکترا [11] و [12] به کاهش محاسبات الگوریتم تطبیقی بالاخص در حیطهٔ شبکه‌های کامپیوتری برای یافتن مدل پرداخته شده است، اما برای بخش بهینه‌سازی در آنها کاری صورت نگرفته است. مقالهٔ [7] که به ارائهٔ یک راه‌حل سخت‌افزاری برای کنترلر پیش‌بین مدل می‌پردازد نیز به صورت کارآمدی محاسبات کنترلر را کاهش نداده است. در مرجع [13] یک پیاده‌سازی سخت‌افزاری برای یک کنترل‌کننده پیش‌بین مدل انجام شده است. در این مقاله برای بهینه‌سازی تابع هدف، از روش برنامه‌ریزی خطی[6] استفاده شده که بسیار ساده و تنها شامل یک ضرب ماتریسی است و البته با توجه به اکثر کاربردها، برنامه‌ریزی خطی گزینه مناسبی نمی‌باشد. از طرفی در این مقاله از روش سنتی و کند درخت جستجوی دودویی[7] استفاده شده است و مؤلفین کاربرد خاصی برای کار و نتیجه آن در نظر نگرفته‌اند.

بر این اساس ما در این مقاله بر مسألهٔ بهینه‌سازی تابع هدف تمرکز یافته و به ارائهٔ یک الگوریتم شبه‌بهینه برای آن می‌پردازیم و همچنین یک معماری سخت‌افزاری کارآمد برای آن معرفی می‌کنیم تا بتوان علاوه بر کاهش محاسبات به افزایش سرعت قابل ملاحظه‌ای نیز دست یافت.

۴- الگوریتم RLS

برای طراحی یک مدل تطبیقی در ابتدا باید یک ساختار مناسب برای مدل انتخاب نمود و سپس با تغییر تعداد پارامترها چندین مدل به دست آورده و برای هر مدل مقادیر پارامترها را حساب کرد و در انتها به ارزیابی پارامترها پرداخت. سیستمی که در این مقاله برای کنترل در نظر گرفته شده است سیستم صف‌های شبکه می‌باشد. در این بخش تنها به توصیف مبانی الگوریتم RLS پرداخته شده و در بخش بعدی شیوه اعمال پیش‌بین مدل را به صف‌های شبکه بیان خواهیم کرد.

با توجه به پویایی بالای مؤلفه‌های موجود در شبکه‌های کامپیوتری، استفاده از ساختار مدل با ورودیهای برون‌زاد[8] پیشنهاد شده است. از مهمترین این ساختارها، ساختار ARX[9] است که در آن از ورودی‌های کنترلی u علاوه بر ورودی‌های ذاتی سیستم (v) استفاده می‌شود:

$$y=\frac{B(z)}{A(z)}u+T(z)v \tag{12}$$

یکی از مدلهای ساختار ARX مدل FIR[10] (طول پاسخ ضربهٔ محدود) می‌باشد و به صورت زیر نوشته می‌شود:

$$\hat{y}_i(n)=\sum_{j=1}^{N}b(j)u_i(n-j) \tag{13}$$

این معادله در شکل ماتریسی به صورت زیر بیان می‌شود:

$$\hat{y}_i(n)=\varphi^T(n)\theta \tag{14}$$

که:

$$\begin{aligned}&\theta^T=[b_1,b_2,...,b_N]\\&\varphi^T(n)=[u(n-1),u(n-2),...,u(n-N)]\end{aligned} \tag{15}$$

در این‌صورت می‌توان تابعی به شکل زیر تحت عنوان "تابع مربع خطا" تعریف کرد و با استفاده از کمینه‌کردن این تابع که در واقع تفاضل بین آنچه مدل بدست آورده با آنچه که واقعاً در خروجی دیده شده است، می‌باشد به تخمین پارامترهای مدل پرداخت:

$$V(\theta,n)=\frac{1}{2}\sum_{j=1}^{N}(y(j)-\varphi(j)\theta)^2=\frac{1}{2}\sum_{j=1}^{N}(\varepsilon^2(j)) \tag{16}$$

با توجه به اینکه در این مرحله از مدل تطبیقی استفاده می‌شود، پارامترهای آن در حین کارکرد سیستم و با جلو رفتن زمان تغییر می‌کنند. این تغییرات در هر گام نسبت به گام قبلی می‌توانند زیاد و یا کم باشند. از آنجایی که سیستم تحت کنترل در این مقاله، شبکه با ترافیک‌ها و توپولوژی‌های متفاوت است، بهترین گزینه استفاده از تخمین پارامترهای مدل با الگوریتم‌های تطبیقی است. الگوریتمی که علاوه برسرعت همگرایی بالا دارای دقت زیادی نیز باشد الگوریتم RLS است که معادلات آن به صورت زیر نوشته می‌شود:

$$\begin{aligned}&\hat{\theta}_i(n)=\hat{\theta}_i(n-1)+\\&\quad P_i(n)\varphi_i(n-1)(y_i(n)-\varphi_i'(n-1)\hat{\theta}_i(n-1))\\&A=\lambda^{-1}(P_i(n-1)-P_i(n-1)\varphi_i(n-1)\varphi_i'(n-1))\\&B=P_i(n-1)(\lambda I+\varphi_i'(n-1)P_i(n-1)\varphi_i(n-1))^{-1})\\&P_i(n)=AB\end{aligned} \tag{17}$$

در این معادلات، $\theta_i(n)$ ماتریس ضرایب سیستم و $\hat{\theta}_i(n)$ نتیجهٔ حاصل از تخمین این ضرایب است. $\lambda\in[0,1]$ نیز پارامتر فراموشی است. λ با توجه به ایستا یا پویا بودن سیستم‌های مختلف، متفاوت است اما در صورت افزایش تعداد تکرارها، انتخاب این پارامتر تأثیر چندانی در مسأله نخواهد داشت.

مشکل اصلی الگوریتم RLS بخصوص اگر درجهٔ سیستم بالا باشد، زیاد بودن حجم محاسبات است. محاسبه P(n) در هر تکرار شامل چندین ضرب ماتریسی و نیز محاسبهٔ ماتریس معکوس می‌باشد. برای افزایش سرعت محاسبات، الگوریتم‌هایی پیشنهاد شده است که از حجم محاسبات الگوریتم در ازای از دست دادن سرعت همگرایی می‌کاهند. در این مقاله روشی ارائه می‌شود که بوسیلهٔ آن می‌توان

بدون از دست دادن سرعت همگرایی و تنها با استفاده از یک روش شبه بهینه، سرعت کار الگوریتم RLS را تا حد زیادی افزایش داد.

۵- اعمال پیش‌بین به زمانبند مسیریاب

هسته‌های اصلی شبکه‌های کامپیوتری، مسیریاب‌هایی هستند که به صورت زنجیره‌ای به یکدیگر متصل شده‌اند و بر اساس الگوریتم مسیریابی خاص هر شبکه، ماشین‌های میزبان را با یکدیگر مرتبط می‌سازند.

یکی از مهمترین بخش‌های یک مسیریاب که نقش مهمی را در تضمین کیفیت سرویس[11] بر عهده دارد بخش زمانبند[12] است که با توجه به وزن‌دهی به صف‌های مختلف و بر اساس الگوریتمهای متفاوت زمانبندی بالاخص WFQ[13] و WF^2Q[14]، بسته‌های موجود در صف‌های مختلف را که بر اساس پارامترهای کیفیت سرویس مرتب شده‌اند، زمانبندی و ارسال می‌کند.

فرض کنید که ورودی کنترل‌کننده، میزان سرویس‌دهی به صف‌های مختلف [6] باشد و خروجی مدنظر از سیستم تأخیر مطلق یا نسبی بسته‌های موجود در هر صف در نظر گرفته شود. از آنجایی‌که در اکثر سیستم‌های کنترل صف به منظور جلوگیری از نوسانی شدن سیستم، مقادیر سرویس‌دهی به صورت کوانتایی تغییر می‌کند، تغییرات ورودی در این مقاله نیز به صورت کوانتایی از پهنای باند در نظر گرفته شده است. بنابراین میزان تغییر در هر ورودی یکی از سه حالت ممکن $\{\delta,0,-\delta\}$ خواهد بود. به عبارت دیگر مقدار هر ورودی فقط یکی از اعضای مجموعه زیر می‌باشد:

$$r_i \in \{0,\frac{1}{s},\frac{2}{s},...,C\} \tag{18}$$

که در آن δ، میزان کوانتایی به شکل زیر است :

$$\delta = \frac{1}{S} \tag{19}$$

C نیز نشان‌دهنده کل پهنای باند موجود می‌باشد. با فرض اینکه تعداد کلاس‌های سرویس برابر با ۴ باشد میزان تغییر در سرویس‌دهی فقط یکی از اعضای مجموعهٔ زیر خواهد بود:

$$\begin{aligned}\Delta r \in \{&(0,0,0,0),(0,0,\delta,-\delta),(0,\delta,0,-\delta),(0,0,-\delta,\delta),(0,-\delta,0,\delta),\\ &(0,\delta,-\delta,0),(0,-\delta,\delta,0),(\delta,0,-\delta,0),(-\delta,0,\delta,0),(\delta,0,0,-\delta),\\ &(-\delta,0,0,\delta),(\delta,-\delta,0,0),(-\delta,\delta,0,0),(\delta,\delta,-\delta,-\delta),(\delta,-\delta,\delta,-\delta),\\ &(\delta,-\delta,-\delta,\delta),(-\delta,\delta,\delta,-\delta),(-\delta,\delta,-\delta,\delta),(-\delta,-\delta,\delta,\delta)\}\end{aligned} \tag{20}$$

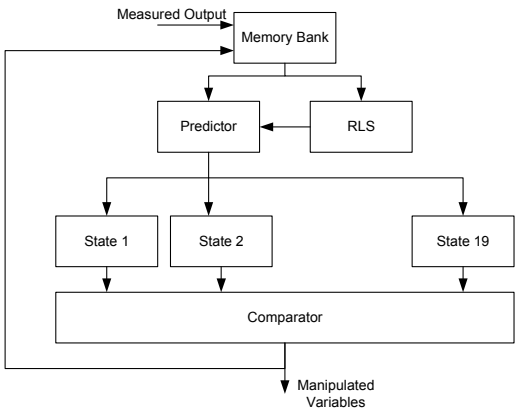
در اینجا باید به این نکته توجه شود که با در نظر داشتن پایا بودن سیستم، در هر لحظه مجموع سرویس‌دهی به صف‌های مختلف که در آنها بسته‌ای در انتظار ارسال موجود است نباید از ظرفیت خط کمتر باشد تا سیستم به صورت بهینه عمل کند. مجموع تغییرات نیز به علت محدودیت پهنای باند نباید بیشتر از ظرفیت خط باشد و بنابراین مجموع تغییرات سرویس‌دهی در هر گام باید برابر صفر باشد. به عبارت دیگر تعداد زیادی از حالت‌های ممکن در مجموعهٔ معرفی شده حذف خواهد شد. به عنوان مثال حالتی که ورودی‌ها به صورت $(\delta,\delta,-\delta,\delta)$ تغییر کنند، مجموع سرویس‌دهی به صف‌های مختلف بعد از اعمال تغییر به اندازه‌ی 2δ بیش از مقدار قبلی است. بنابراین در گام جدید باید به اندازه $C+2\delta$ به صفها سرویس داد که از لحاظ فیزیکی به علت محدودیت پهنای باند غیرممکن است. بنابراین با حذف حالت‌هایی که مجموع مولفه‌های آنها صفر نیست، در مجموع می‌توان ۱۹ حالت را برای تغییر خروجی در گام بعدی متصور شد. نکته‌ای که الگوریتم پیشنهادی را از حالت بهینه به شبه بهینه تبدیل می‌کند جستجوی تابع هدف تنها به ازای تمام این ۱۹ حالت بجای جستجو در کل فضای حالت است.

به صورت کلی، الگوریتم‌های مرتبط با تضمین کیفیت سرویس سعی می‌کنند تا عدالت[15] را بین صف‌های مختلف برقرار کنند و در عین حال چون تأخیر برای بعضی از صفها از اهمیت کمتری نسبت به دیگران برخوردار است، باید با آنها به‌گونه‌ای برخورد شود که کار آنها نیز مختل نگردد. الگوریتم AMPCS [5] مناسبترین الگوریتم ارائه شده برای تضمین کیفیت سرویس با مشخصات فوق و ضمن استفاده از کنترلرهای MPC است.

یکی از مشکلاتی که بر سر راه پیاده‌سازی الگوریتم AMPCS وجود دارد، حجم بالای محاسبات آن است [5]. در واقع یکی از دلایلی که باعث محدود شدن کاربرد کنترلرهای پیش‌بین مدل شده است نیز همین نکته می‌باشد. در نتیجه برای استفاده از این الگوریتم راه‌حلهایی پیشنهاد شده است تا بتوان از آن در محیط‌های پرسرعت نظیر شبکه‌های کامپیوتری بهره جست. یکی از این روش‌ها ارائهٔ یک معماری خاص برای پردازنده مرکزی است تا محاسبات پیش‌بین را در سیکلهای کمتری انجام دهد. راه حل ارائه شده در این مقاله الگوریتم نیمه بهینه SMPCS[16] است.

۶- پیاده‌سازی الگوریتم SMPCS

برای اینکه الگوریتم SMPCS را بتوان در سرعتهای بالا پیاده‌سازی نمود از چند واحد پردازش موازی برای انجام عملیات بهینه‌سازی استفاده می‌شود. نکته‌ای که باید به آن توجه گردد این است که بر خلاف تمام پیاده‌سازی‌های پیشین که در آنها از پردازنده و نرم‌افزار برای اجرای الگوریتم کنترل‌کننده‌ی پیش‌بین مدل استفاده شده است، در این مقاله یک معماری جدید معرفی می‌گردد که برای افزایش کارآیی بخش اعظم محاسبات به صورت سخت‌افزاری انجام می‌شود. البته بخشی از الگوریتم که بار محاسباتی چندانی ندارد و نیز پیاده‌سازی آن به صورت کاملاً سخت‌افزاری هزینه بالایی در بر خواهد داشت نیز می‌تواند بر عهدهٔ یک پردازندهٔ معمولی گذاشته شود.

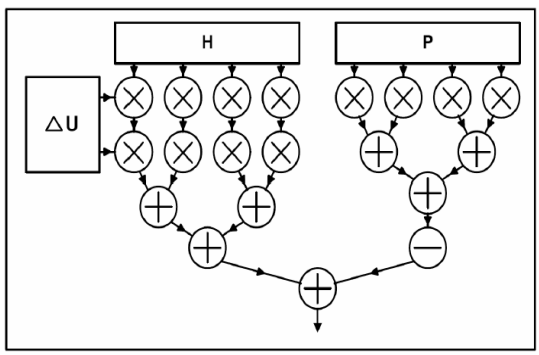


شکل ۲- معماری SMPCS

همانطور که گفته شد، یک ماتریس H در انتهای کار توسط بخش پیش‌بین تولید می‌شود.. بعلاوه یک ماتریس P نیز توسط الگوریتم RLS تولید می‌گردد. سپس بخش بهینه‌سازی تابع هدف با دریافت این دو ماتریس و با بکارگیری مجموعهٔ تغییرات ورودی تابع زیر را به کمک سخت‌افزار پیاده‌سازی می‌کند:

$$\Delta U^T H \Delta U - P\Delta U$$

در واقع این تفاضل، تفاضل بین مقدار بدست آمده از مدل سیستم (الگوریتم RLS) و مقدار بدست آمده از اندازه‌گیری‌های کنترلر است که هر چه دارای مقدار کوچکتری باشد بهتر است. از آنجاکه این حاصل باید به ازای تمام ۱۹ حالت از تغییرات ورودی که قبلاً ذکر شد محاسبه و در نهایت جواب آنها مقایسه شود برای این منظور به یک مقایسه‌گر نیاز خواهد بود.



شکل ۳- عملیات ضرب و جمع بر روی ماتریس P و H

۷- نتایج آزمایشات

برای پیاده‌سازی معماری SMPCS از بلوکهای سخت‌افزاری انجام عملیات ریاضی سریع [4] استفاده شده است. این بلوکها شامل ضرب‌کننده مبنای ۴ الگوریتم Booth است که با استفاده از یک جمع‌کننده سریع چند عملوندی CSA[17] عمل ضرب را انجام می‌دهد. بعلاوه جمع‌کننده نیز از نوع جمع‌کنندهٔ CLA[18] می‌باشد. همچنین برای پیاده‌سازی مقایسه‌کننده چندعملوندی، از یک آرایش خط لوله‌ای سریع با پنج سطح مقایسه برای یافتن مقدار کمینه‌کننده تابع هدف از بین ۱۹ خروجی هر حالت استفاده شده است. برای پیاده‌سازی بر روی FPGA از قطعهٔ Spartan 3 استفاده شده است. نتایج پیاده‌سازی قسمتهای مختلف در جدول ۱ نشان داده شده است.

جدول ۱- نتایج پیاده‌سازی الگوریتم بر روی سخت‌افزار

بلوکهای سخت‌افزاری	تعداد Slice	مقداربدترین تاخیر (ns)
ضرب‌کننده 16X16	۲۹۵	۲۵
جمع‌کننده CLA	۵۶	۱۵
مقایسه‌کننده	۶۸۵	۶۶
هر یک از ۱۹ حالت	۳۹۵۷	۹۶
مکمل ۲ کننده	۲۵	۱۵
کل بلوک برنامه‌ریزی مربعی	۷۵۸۶۸	۱۴۵

برای مقایسهٔ بین پیاده‌سازی سخت‌افزاری و نرم‌افزاری عملیات بهینه‌سازی تابع هدف و در واقع داشتن معیاری مناسب برای قابل قبول بودن پیاده‌سازی سخت‌افزاری، کل عملیات بلوک برنامه‌ریزی مربعی توسط زبان برنامه‌نویسی LabView نیز انجام شده است. لازم به ذکر است که نرم‌افزار LabView بر روی یک کامپیوتر با پردازندهٔ Pentium IV با فرکانس ۲،۸ گیگاهرتز و با ۱ گیگابایت حافظهٔ RAM و حافظه Cache معادل ۵۱۲ کیلو بایت اجرا شده است. نتیجهٔ مقایسهٔ زمان اجرای سخت‌افزار و نرم‌افزار در جدول ۲ آمده است:

جدول ۲- نتایج مقایسهٔ سرعت اجرا در سخت‌افزار و نرم‌افزار

	سخت‌افزار	نرم‌افزار
زمان بهینه‌سازی	۱۴۵ نانوثانیه	۴۴ میلی ثانیه

به منظور ارزیابی معماری ارائه شده برای استفاده در شبکه‌های کامپیوتری، در شبیه‌سازیها از یک پالس ساعت با پهنای ۱۵ نانوثانیه استفاده شده است. با توجه به نوع پیاده‌سازی سیستم و نیز توجه به طول خط لوله در قسمتهای مختلف، هر عمل ضرب در ۲ پالس ساعت، هر عمل مقایسه در ۵ پالس ساعت به ازای هر سطح مقایسه و در نهایت کل عملیات بهینه‌سازی در ۱۰ پالس ساعت صورت می‌گیرد. با این توصیف و برای استفاده از کنترلر پیش‌بین مدل در زمانبند شبکه‌ها، می‌توان با در نظرگرفتن ۴ پالس ساعت برای عملیات الگوریتم RLS، کلاً با ۱۴ پالس ساعت عملیات بهینه‌سازی را انجام داد که با سرعتهای کنونی شبکه سازگاری ندارد. اما با اعمال تغییرات ورودی در صفحه‌های شبکه‌های کامپیوتری - مثلاً تغییر میزان سرویس‌دهی هر چند پالس ساعت یکبار- می‌توان به سادگی از سیستم موجود در زمانبند بهره برد. نکتهٔ مهمی که باید به آن توجه شود این است که عدم تغییرات سریع در میزان سرویس‌دهی به منظور جلوگیری از نوسانی شدن شبکه به هنگام تغییرات جزیی در وضعیت لینک‌های مختلف کاملاً منطقی است و بنابراین معماری ارائه شده به خوبی می‌تواند در زمان‌بند شبکه‌های کامپیوتری مورد استفاده قرار گیرد.

۸- نتیجه‌گیری

در این مقاله یکی از ابزارهای مهم کنترل مدرن یعنی کنترلر پیش‌بین مدل با استفاده از ارائهٔ یک معماری سخت‌افزاری به منظور افزایش سرعت محاسبات، پیاده‌سازی و ارزیابی شد. به منظور افزایش سرعت پیاده‌سازی و نیز جلوگیری از کاهش دقت در پاسخ‌ها یک سیستم شبه‌بهینه پیشنهاد و ارزیابی گردید. نتایج پیاده‌سازی نشان می‌دهد که در مقایسه با پیاده‌سازی‌های معمول نرم‌افزاری موجود، معماری معرفی شده به افزایش سرعت چشم‌گیری دست‌یافته است که با افزایش سرعت این کنترلرها می‌توان از آنها در طیف وسیعی از کاربردها استفاده کرد.

برای بهبود طرح و انجام کارهای آتی مسائل مختلفی می‌تواند مطرح شود. استفاده از سیستم اعداد لگاریتمی بجای استفاده از سیستم اعداد معمولی برای بهبود حجم سخت‌افزار یکی از کارهای مهمی است که می‌تواند صورت گیرد. به این ترتیب با محدودسازی اعداد بزرگ و نیز با استفاده از جمع به جای ضرب، سرعت کار افزایش خواهد یافت. همچنین می‌توان با بهره‌گیری از یک مدل غیرخطی برای سیستم به جای مدل خطی متغیر با زمان و نیز استفاده از یک جدول داده جستجو به سرعت انجام محاسبات این کنترلرها افزود.

سپاسگزاری

در اینجا لازم است مراتب تشکر و قدردانی نویسندگان این مقاله را از آقای دکتر مهران محرمیان معلم ابراز نمائیم که در تمام مراحل کار ما را یاری نمودند و از هیچ کمکی برای به ثمر رسیدن این کار دریغ نکردند و به حق برای این مقاله کمتر از مؤلفین آن تلاش ننمودند.

مراجع

[1] J.Richalet, A. Rault, J. L.Testud and J.Papon, *"Model Predictive Heuristic Control,"* Automatica, Vol. 14, pp. 413-428, 1978.

[2] Minyue Fu, Zhi-Quan Luo, Yinyu Ye, *"Approximation Algorithms for Quadratic Programming,"* Journal of Combinational Optimization, Vol. 2, No. 1, pp. 29- 50, 1998.

[3] Petter Tondel, Tor A. Johansen, *"Complexity Reduction in Explicit Linear Model Predictive Control,"* 15th Control International Federation of Automatic World Congress, 2002.

[4] Petillon, T. Gilloire, A. Theodoridis, S. Cnet Lannion, *"Complexity Reduction in Fast RLS Transversal Adaptive Filters with Application to Acoustic Echo Cancellation,"* IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 37-40, 1992.

[5] Israel Koren, *"Computer Arithmetic Algorithms",* Textbook, A K Peters, Ltd, Natick Massachusetts, 2nd Edition, 2002.

[6] M. Mahramian, M. Taheri, H. Haeri, *"AMPCS: Adaptive Model Predictive Control Scheduler,"* submitted to International Journal of Communication Systems, January 2006.

[7] M. Mahramian, H. Taheri, *"A Model Predictive Control Approach for Guaranteed Proportional Delay in DiffServ Architecture,"* Proceedings of the 10th Asia-Pacific Conference on Communications, pp. 592-597, August, 2004.

[8] A. P. Poloski, and J. C. Kantor *"Application of Model Predictive Control to Batch Processes,"* Computers and Chemical Engineering, Vol. 27, pp. 913-926, 2003.

[9] L.G. Bleris, J. Garcia, M. V. Kothare, M. G. Arnold, *"Towards Embedded Model Predictive Control for System-on-Chip Application,"* Journal of Process Control, Vol. 16, No. 3, pp. 255-264, March 2006.

[10] F. Di Palma, and L. Magni, *"On Optimality of Nonlinear Model Predictive Control,"* Information and System Department, 2003.

[11] H. Khanmirza, S. Zarifzadeh, N. Yazdani, *"ADPQ:an Adaptive Approach for Expedited Forwarding Traffic Scheduling,"* 10th IEEE Symposium on Computers and Communications, pp. 801-806, 2005.

[12] A.H. Mohsenian Rad, M. Haeri. *"Adaptive Model Predictive TCP Delay-based Congestion Control Algorithm,"* Accepted Paper to Publish in Computer Communications, 2006.

[13] Tor A. Johansen, Warren Jackson, Robert Schrieber, Petter Tondel, *"Hardware Architecture Design for Explicit Model Predictive Control,"* American Control Conference, July, 2005.

زیرنویس‌ها

[1] Proportional Integral Differential
[2] Model Predictive Controllers
[3] Closed-loop
[4] Quadratic programming
[5] Recursive Least Squares Algorithm
[6] Linear Programming
[7] Binary Search Tree
[8] Exogenous
[9] Auto Regressive exogenous
[10] Finite impulse response
[11] Quality of Services
[12] Scheduler
[13] Weighted Fair Queuing
[14] Worst case fair weighted fair queuing
[15] Fairness
[16] Sub-Optimal model predictive controller scheduling
[17] Carry Save Adder
[18] Carry Look Ahead