۱۳۰۷
دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده مهندسی برق و کامپیوتر

**پایان‌نامه دوره کارشناسی ارشد مهندسی برق–کنترل**

# طراحی سیستم‌های کنترل پیش بین سریع با استفاده از روش‌های پیشرفته‌ی بهینه سازی

**توسط:**

**سامان سیروس**

**استاد راهنما:**

**دکتر علی خاکی صدیق**

**استاد مشاور:**

**دکتر محمدرضا پیغامی**

**زمستان ۱۳۹۱**

بسم الله الرحمن الرحیم

# تأییدّه هیات داوران

(برای پایان نامه)

اعضای هیئت داوران، نسخه نهائی پایان نامه خانم / آقای:

را با عنوان:

از نظر فرم و محتوی بررسی نموده و پذیرش آن را برای تکمیل درجه کارشناسی/ کارشناسی ارشد تأیید می‌کند.

| اعضای هیئت داوران | نام و نام خانوادگی | رتبه علمی | امضاء |
| --- | --- | --- | --- |
| ۱- استاد راهنما | دکتر علی خاکی صدیق | استاد | |
| ۲- استاد مشاور | دکتر محمدرضا پیغامی | دانشیار | |
| ۳- استاد مشاور | | | |
| ۴- استاد ممتحن | دکتر محمدعلی نکوئی | دانشیار | |
| ۵- استاد ممتحن | دکتر وحید جوهری مجد | دانشیار | |
| ۶- نماینده تحصیلات تکمیلی | دکتر محمدعلی نکویی | دانشیار | |

**تقدیم**

**پدر و مادر عزیزم**

**و**

**اشکان و پیمان نازنینم**

# تشکر و قدردانی

در ابتدا باید از پدر و مادر عزیزم که در طول این سال‌ها همیشه در کنارم بوده‌اند تشکر و قدردانی کنم، اگر نبود حمایت‌هایشان و صبر و بردباریشان حرکت برایم ممکن نبود. همچنین بر خود لازم می‌دانم از برادران عزیزم اشکان و پیمان که برادری کردند نهایت قدردانی را بنمایم.
همچنین جا دارد از آقای دکتر خاکی صدیق که در طول دوره‌ی کارشناسی ارشد راهنمایم بودند صمیمانه تشکر کنم. در اینجا می‌باید از جناب آقای دکتر محمدرضا پیغامی استاد مشاور عزیزم در دانشکده علوم دانشگاه خواجه نصیر تشکر کنم که نه تنها در ریاضیات راهنمایم بودند که برایم آموزگار اخلاق بودند، آموختند آنچه گفتنی نبود و در کنارم بودند بیش از آن چه انتظار داشتم. بعلاوه بر خود لازم میدانم مراتب احترام و تشکر را از جناب آقای دکتر مازیار صلاحی استاد محترم گروه ریاضی کاربردی دانشکده علوم ریاضی دانشگاه گیلان که وقت خود را در اختیارم گذاشتند و از حمایتم دریغ نکردند و همچنین از پروفسور Stephen Wright استاد دانشکده علوم کامپیوتر دانشگاه ویسکانسین در مدیسون به خاطر راهنمایی‌هایشان و همچنین کمک در نوشتن کد متلب نهایت تقدیر و تشکر را داشته باشم.
از یکایک دوستانم در گروه کنترل و بخصوص در آزمایشگاه کنترل پیشرفته که هر کدام به نحوی مرا در طول تحصیل و همچنین در انجام این پروژه یاری رساندند نهایت قدردانی را می‌نمایم. دوستان عزیزم سرکار خانم دکتر برزگر و آقای مهندس علی انصاری ورودی‌های گروه سیستم در طول این سال‌ها رفاقت را در حق من تمام کردند. همچنین دوستان عزیزم علیرضا برزگر، محمد جهوانی، خانم دکتر انسیه نوبختی، پیمان باقری، مجتبی نوری منظر، عطاالله گوگانی خیابانی حق بزرگی بر گردنم دارند. برای تمامی این دوستان خوشبختی و شادباش را آرزومندم.

# چکیده

در این پایان نامه مسئله‌ی بهینه سازی در کنترل پیش بین مورد بررسی قرار گرفته است. در کنترل پیش بین، برای یافتن سیگنال کنترلی، یک مسئله‌ی بهینه سازی درجه دوم بایستی حل شود و از آنجا که این مسئله مقید است روش‌های بهینه سازی مقید درجه دوم برای حل آن قابل ارائه است. برای حل مسئله‌ی بهینه سازی که در کنترل پیش بین با آن مواجه هستیم، روش‌های گوناگونی وجود دارد. در این پایان نامه به یک الگوریتم خاص به نام الگوریتم پیش‌گو-اصلاح‌گر مِهروترا پرداخته شده است و پس از شرح مزیت‌ها و ویژگی‌های آن، یک گونه‌ی اصلاح شده از این الگوریتم، ابتدا برای مسئله‌ی برنامه ریزی درجه دوم توسعه داده شده و سپس به مسئله‌ی کنترل پیش بین اعمال گردیده است. در نهایت با در نظر گرفتن چند سیستم مختلف و اعمال کنترل پیش بین پیشنهادی به آن‌ها در حالت‌های مختلف، با افق‌های پیش بین گوناگون، کارآمدی این روش به دقت بررسی شده است. برای بررسی کارآمدی این روش، سرعت پاسخ‌دهی آن با دستور مخصوص برنامه ریزی درجه دوم نرم افزار متلب مقایسه شده است. نتایج حاصل به خوبی نشان دهنده‌ی سرعت بیشتر در حل مسئله‌ی کنترل پیش بین در هر گام بوده است.



**کلید واژه:** کنترل پیش بین، بهینه سازی مقید، الگوریتم بهینه سازی درجه دوم نقطه درونی پیش‌گو-اصلاح‌گر مهروترا، الگوریتم بهینه سازی نقطه درونی اصلاح شده‌ی مهروترا.

## فهرست مطالب

**عنوان** **صفحه**

# فهرست جدول‌ها

**عنوان** **صفحه**

# فهرست شکل‌ها

**صفحه** **عنوان**

# فهرست علایم و نشانه‌ها

| عنوان | علامت اختصاری |
|---|---|
| فضای بردارهای n بعدی | $\mathbb{R}^n$ |
| نرم اقلیدسی | $\|\cdot\|$ |
| مجموعه‌ی شدنی اولیه-دوگان $\{(x,\lambda,s) \mid Ax=b, A^T\lambda+s=c, (x,s)\geq 0\}$ | $\mathcal{F}$ |
| مجموعه‌ی اکیداً شدنی اولیه-دوگان $\{(x,\lambda,s) \mid Ax=b, A^T\lambda+s=c, (x,s)>0\}$ | $\mathcal{F}^0$ |
| معیار دوگانی، تعریف شده به صورت $x^Ts$ | $\mu$ |
| مانده‌ی اولیه، تعریف شده به صورت $Ax-b$ | $r_b$ |
| مانده‌ی دوگان، تعریف شده به صورت $A^T\lambda+s-c$ | $r_c$ |
| تکرار عمومی روش‌های اولیه=دوگان نقطه‌ی درونی | $(x^k,\lambda^k,s^k)$ |
| گام عمومی اولیه-دوگان | $(\Delta x,\Delta\lambda,\Delta s)$ |
| ژاکوبین $F$ | $J(.)$ |
| تابعی از $(x,\lambda,s)$ شامل معادلات شرایط $KKT$ | $F(.)$ |
| پارامتر طول گام | $\alpha,\alpha_k$ |
| پارامتر مرکزیت برای محاسبات گام | $\sigma$ |
| همسایگی نُرم دو مسیر مرکزی | $\mathcal{N}_2(\theta)$ |
| همسایگی نُرم بینهایت مسیر مرکزی | $\mathcal{N}_{-\infty}$ |

# فصل ۱- مقدمه

## ۱-۱- پیشگفتار

کنترل پیش بین یکی از الگوریتم‌های کنترلی است که جایگاه خوبی برای خود در صنایع فراهم کرده است. همچنین علیرغم سابقه‌ی طولانی استفاده از آن و تحقیقات بسیاری که در این حوزه صورت گرفته است، همچنان به دلیل ویژگی‌های خاصی نظیر توانایی بیان قیدهای سیستم به صورت صریح، در محافل صنعتی و دانشگاهی مورد توجه است. استفاده از کنترل پیش بین نیازمند حل یک مسئله‌ی بهینه سازی است. در مسائل ابعاد وسیع، مدت زمانی که برای حل این مسئله‌ی بهینه سازی صرف می‌شود، نسبت قابل توجهی از کل زمان لازم برای محاسبه سیگنال کنترل است. این ضعف سبب محدودیت استفاده از این کنترل کننده در صنایعی با زمان نمونه برداری سریع شده است. تلاش‌های زیادی در این رابطه در طی سالیان گذشته صورت گرفته است تا از میزان زمان لازم برای حل مسئله بهینه سازی کاسته شود. کنترل کننده های صنعتی‌ای که ارائه شده‌اند نیز از رویکردها و روش‌های گوناگون برای رفع این نقص استفاده می‌کنند، اما با تمام این تلاش‌ها هنوز هم شاید بتوان گفت رایج‌ترین کاربرد کنترل کننده های پیش بین، در کنترل فرآیندها است که در آن دینامیک سیستم‌ها ذاتاً کُند است. بنابراین ارائه‌ی یک الگوریتم ریاضی نوین که باعث بهبود سرعت عملکرد کنترل کننده های پیش بین شود می‌تواند اثر شگرفی در افزایش کاربرد این کنترل کننده در صنایع داشته باشد.

## ۱-۲- اهمیت کنترل پیش بین

کنترل پیش بین یکی از موفق‌ترین الگوریتم‌های کنترلی ارائه شده طی چند دهه‌ی گذشته است که جایگاه وسیعی در صنایع مختلف و بویژه در کنترل فرآیندها پیدا کرده است. این تکنیک یک الگوریتم تکرارشونده است که در هر گام آن، یک مسئله‌ی کنترل بهینه‌ی حلقه باز حل می‌شود و این کار به صورت تکرارشونده ادامه دارد[۲،۱]. اگرچه کنترل پیش بین در ابتدا برای رفع نیازهای کنترلی خاص در نیروگاه‌ها و پالایشگاه‌های نفت بوجود آمد، امروزه آن‌را می‌توان در صنایع گوناگونی از جمله صنایع غذایی، شیمیایی، خودروسازی و صنایع هوافضا یافت [۳].

در هر گام کنترل پیش بین، یک مسئله کنترل بهینه با اهداف بولزا[1] حل می‌شود و پاسخ آن به سیستم اعمال می‌شود تا هنگامی‌که یک نمونه‌گیری دیگر از سیستم بدست آید.[۴] حال اطلاعات

[1] Bolza objective

سیستم جدیدِ به روز شده برای فرموله کردن یک مسئله‌ی جدید کنترل بهینه بکار می‌رود. بنابراین از سیستم به مدل، فیدبک بوجود می‌آید و این رویه دائماً تکرار می‌گردد.

مزیت کنترل پیش بین توانایی آن در برشمردن صریح قیود ورودی و قیود حالت‌ها، توانایی آن در اِعمال به مسائل کنترل چند ورودی-چند خروجی و ماهیت آن در حل برخط[۱] مسئله است (که در زمان‌هایی که مسئله را نمی‌توان به صورت خارج از خط[۲] حل کرد کاربرد دارد). یکی از نقاط ضعف این تکنیک هزینه‌ی محاسباتی بسیار زیاد آن است که کاربرد کنترل پیش بین را محدود به سیستم‌های با دینامیک کُند کرده است. برای مسائلی مانند کنترل پیش بین، مسئله‌ی کنترل بهینه ای که باید در هر گام حل شود یک مسئله‌ی درجه دوم محدب[۳] است[۲،۴].

وقتی که مسئله بدون قید است، در واقع کنترل پیش بین با افق نامحدود همان مسئله‌ی[۴] LQR است. حتی اگر در مسئله کنترل پیش بین قید هم داشته باشیم، اگر مسئله دارای افق نامحدود باشد می‌توان آن‌را با چند گام به مسئله‌ی LQR تبدیل نمود. (برای این منظور[۵،۶،۷] را ببینید)

کنترل پیش بین روشی جدید در طراحی کنترل کننده‌ها نیست. در واقع کنترل پیش بین مسئله کنترل بهینه استاندارد را حل می‌کند (با این تفاوت که در کنترل پیش بین- بر خلاف کنترل بهینه خطی $H_2$ و $H_\infty$ که افق نامحدود دارد - مسئله کنترل بهینه باید افق محدود داشته باشد). تفاوت اصلی آن با سایر روش‌های کنترل بهینه این است که کنترل پیش بین به صورت برخط مسئله را برای حالت فعلی حل می‌کند، در حالی که در سایر روش‌های کنترل بهینه فیدبک به صورت خارج از خط محاسبه می‌شود و کنترل بهینه را برای تمام حالت‌ها بدست می‌آورد[۱].

## ۱-۳- مسئله‌ی بهینه سازی در کنترل پیش بین

همان طور که گفته شد در مسئله‌ی کنترل پیش بین خطی، در هر گام برای یافتن مقدار سیگنال کنترلی مطلوب، باید یک مسئله‌ی بهینه سازی مقید درجه دوم حل شود (اگرچه تلاش‌هایی برای مدل سازی کنترل پیش بین به صورت برنامه ریزی خطی صورت گرفته است اما به دلیل مشکلاتی که این نوع برخورد با مسئله ایجاد می‌کند این رویکرد چندان رایج نیست، برای مثال به[۲] مراجعه شود. همچنین برای اطلاع از کنترل کننده های صنعتی که از برنامه ریزی خطی برای بهینه سازی در کنترل پیش بین استفاده می‌کنند به انتهای همین فصل مراجعه شود). از آنجایی که حل این مسئله‌ی بهینه سازی

1 Online
2 Offline
3 Convex quadratic program
4 Linear Quadratic Regulator

زمان‌گیر است، اِعمال کنترل پیش بین به صنایع با زمان نمونه برداری سریع را غیرممکن ساخته است. از این رو پژوهش‌های زیادی در جهت به‌کارگیری الگوریتم‌های سریع بهینه سازی در کنترل پیش بین صورت گرفته است. در[۸] سعی شده است که مسئله به صورت اسپارس[۱] حل شود، در [۹،۴] از روش‌های نقطه درونی برای حل مسئله بهینه سازی استفاده شده است.[۱۱،۱۰] از روش‌های مجموعه فعال[۲] برای حل مسئله بهینه سازی استفاده کرده‌اند؛ همچنین برخی روش‌های هوشمند هم برای حل مسئله بهینه سازی در کنترل پیش بین استفاده شده است[۱۳،۱۲]. معروف‌ترین روش‌های کلاسیک حل مسائل بهینه سازی درجه دوم روش‌های نقطه درونی[۳] و روش‌های مجموعه فعال هستند، این دو دسته روش در [۱۵،۱۴] با هم مقایسه شده‌اند.

همچنین تلاش‌هایی هم برای گریز از حل مسئله بهینه سازی در کنترل پیش بین صورت گرفته است. از جمله محبوب‌ترین این تلاش‌ها، حل مسئله به صورت خارج از خط برای کل فضا و ایجاد یک جدول مرجع[۴] برای کل فضا است. سپس مسئله کنترل پیش بین به صورت برخط حل می‌شود و هنگام برخورد با مسئله بهینه سازی، تنها با مراجعه به این جدول مقدار بهینه برداشت می‌شود[۱۶]. در [۱۷] گفته شده است که با این روش، سرعت الگوریتم یک صدبار نسبت به استفاده از یک الگوریتم بهینه سازی عمومی افزایش می‌یابد. اما برخی مقالات مانند [۱۱] این رویکرد را با شک مواجه کرده‌اند و استفاده از بهینه سازهای معمول و رایج را راه حل مطمئنی دانسته‌اند. همچنین برخی تلاش‌ها مانند[۱۸] مبحثی را تحت عنوان کنترل پیش بین صریح[۵] توسعه داده‌اند که در آن‌ها تعداد زیادی مسئله برنامه ریزی درجه دوم به صورت خارج از خط برای تمامی حالات اولیه سیستم حل می‌شود و سپس یک تابع صریح با استفاده از حل این مسئله های برنامه ریزی درجه دوم مطرح می‌شود. ابعاد این جدول ممکن است با افزایش تعداد حالت‌ها، ورودی‌ها و افق پیش بینی به صورت نمایی گسترش پیدا کند، بنابراین محدودیت کنترل پیش بین صریح معمولاً حافظه کامپیوتری مورد نیاز است و کاربرد آن به سیستم‌های با ابعاد کوچک محدود می‌شود[۱۶].

از آنجا که مسئله‌ی بهینه سازی در کنترل پیش بین یک الگوریتم تکرارشونده است، می‌توان از اطلاعاتی که از تکرارهای پیشین در حل مسئله بهینه سازی بدست آمده است برای حل مسئله بهینه سازی استفاده کرد، برای اطلاعات بیشتر و کاربرد آن در کنترل پیش بین به [۲۱،۲۰،۱۹] مراجعه شود.

---

[1] Sparse
[2] Active-set methods
[3] Interior-point methods
[4] Lookup table
[5] Explicit Model predictive control

همچنین برای اطلاعات بیشتر در مورد تئوری ریاضی این کار که به شروع گرم[1] شناخته می‌شود به [۲۲] مراجعه شود. این کار یکی از روش‌های رایج در مبحث بهینه سازی در ریاضیات کاربردی است و در صورتی که در هر تکرارِ مسئله‌ی بهینه سازی فرض کنیم که فضای مسئله‌ی بهینه سازی تغییر نمی‌کند، قابلیت اجرا دارد.

## ۱-۴- بهینه سازی و روش‌های جدید

همان‌طور که گفته شد مسئله‌ی بهینه سازی که در کنترل پیش بین با آن مواجه هستیم، مسئله بهینه سازی درجه دوم است. برای حل مسئله بهینه سازی درجه دوم الگوریتم‌های بهینه سازی متفاوتی ارائه شده است که می‌توان موارد زیر را برشمرد:

۱- روش‌های نقطه درونی[۲۴،۲۳]
۲- روش‌های مجموعه فعال[۲۳]
۳- لاگرانژین افزوده[2][۱۶،۲۵]
۴- گرادیان الحاقی[3][۲۳]
۵- تصویر گرادیان[4][۲۳]
۶- بسط روش سیمپلکس برای مسئله برنامه ریزی درجه دوم[۲۷،۲۶]

این روش‌های بهینه سازی دارای خواص متفاوتی هستند که کاربرد آن‌ها را از هم جدا می‌کند. در سال‌های اخیر توجه به روش‌های نقطه درونی و مجموعه فعال بیشتر شده است، در سال ۱۹۸۴ کارمارکر[5] الگوریتمی جدید ارائه کرد [۲۸] که باعث تحول شگرفی در زمینه بهینه سازی خطی شد. اگرچه الگوریتم کارمارکر برای مسائل برنامه ریزی خطی توسعه یافته بود، اما به سرعت کاربرد خود را در برنامه ریزی درجه دوم هم باز کرد، این دسته روش‌ها که با نام روش‌های نقطه درونی شناخته می‌شوند، امروزه هنوز هم با اقبال بسیاری برای کاربردهای مهندسی مواجه هستند. مزیت اصلی روش‌های نقطه درونی این است که برخلاف روش‌های پیشین که دارای پیچیدگی محاسباتی از درجه نمایی بودند، روش‌های نقطه درونی دارای پیچیدگی محاسباتی از درجه چندجمله‌ای هستند، به این معنا که با افزایش ابعاد مسئله، زمان لازم برای الگوریتم برای یافتن نقطه‌ی بهینه به صورت چندجمله‌ای افزایش می‌یابد؛

---

1 Warm start
2 Augmented Lagrangian
3 Conjugate gradient
4 Gradient projection
5 Karmarker

حال آن‌که الگوریتم‌هایی مانند سیمپلکس و مجموعه فعال دارای این خاصیت هستند که با افزایش ابعاد مسئله پیچیدگی سیستم به صورت نمایی افزایش می‌یابد (برای مثال[۲۹] را ببینید).

در این پایان نامه به علت مطلوبیتی که سرعت الگوریتم بهینه سازی داشته است به سراغ روش‌های نقطه درونی رفته‌ایم. اساس کار روش‌های نقطه درونی، استفاده از تکنیک‌های کمینه سازی هموار[1] (معمولاً روش نیوتون) برای حل یک سری مسائل نامقید (یا با قیدهای مساوی) هموار است. عبارت "نقطه درونی" به این دلیل به این روش‌ها نسبت داده می‌شود که تکرارهای این الگوریتم همگی شدنیِ مطلق[2] هستند، یعنی قیدها را ارضا می‌کنند، بنابراین در "درون" مجموعه شدنی قرار دارند[۳۰]. روش‌های نقطه درونی به زیرشاخه های:

الف) روش‌های حامل[3]

ب) روش‌های مسیریاب[4]

تقسیم می‌شوند[۳۱].

روش‌های مسیریاب به خوبی مورد بحث و قبول واقع شده‌اند و در عمل هم پیاده‌سازی شده‌اند. در این پایان نامه تمایل به استفاده از یک الگوریتم خاص از زیر مجموعه روش‌های مسیریابِ نقطه درونی است. روش‌های مسیریاب به زیر مجموعه های:

۱- با گام کوتاه[5]
۲- با گام بلند[6]
۳- پیش گو- اصلاح گر[7]

تقسیم می‌شوند[۳۲]. در میان روش‌های پیش گو-اصلاح گر، الگوریتم مهروترا[8] از نظر توانایی پیاده‌سازی از همه کارآمدتر بوده است[۳۲،۲۳،۳۳].

اگرچه الگوریتم پیش‌گو-اصلاح‌گر مهروترا در عمل بسیار موفق بوده است و بسته های نرم افزاری گوناگون بر اساس این الگوریتم ساخته شده است (برای مثال به[۳۴،۳۵،۳۶،۲۴،۳۷،۳۸،۳۹] مراجعه شود)، اما تلاش‌هایی هم در جهت اصلاح آن ارائه شده است که از آن جمله الگوریتم ارائه شده توسط

[1]Smooth
[2]Strictly feasible
[3]Barrier methods
[4]Path-Following methods
[5]Short-step methods
[6]Long-step methods
[7]Predictor-Corrector Methods
[8]Mehrotra

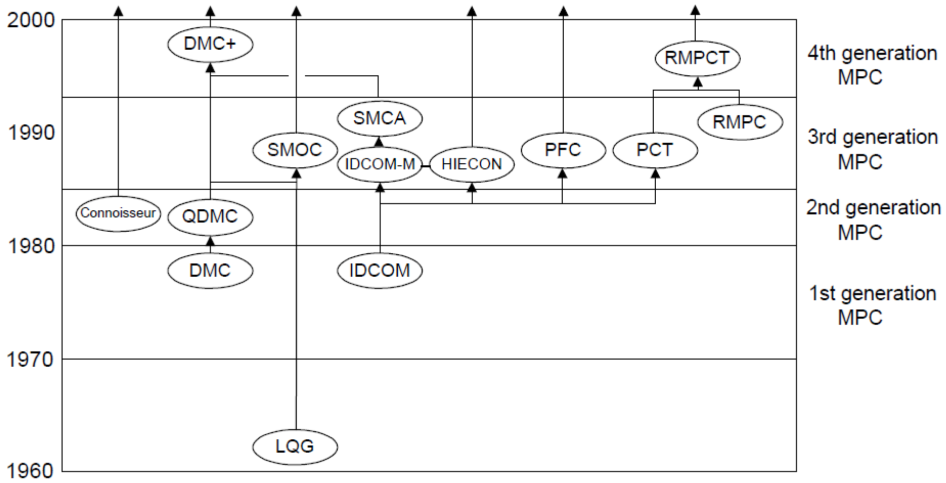
صلاحی-ترلکی[1] است[۴۱،۴۰]. در این پایان نامه تلاش شده است که هم برای الگوریتم پیش گو-اصلاح‌گر مهروترا و هم برای الگوریتم اشاره شده در مرجع [۴۱،۴۰] برنامه‌ی متلب[2] نوشته شود. در انتها نیز، نتیجه با دستور بهینه سازی درجه دوم جعبه ابزار بهینه سازی (ویرایش ۵،۰) نرم افزار متلب مقایسه شده است.

## ۱–۵– نرم افزارهای تجاری کنترل پیش بین

همان طور که گفته شده کنترل پیش بین در صنعت کاربرد فراوان دارد، از این روی در این بخش ضمن معرفی برخی از شرکت‌های فعال در زمینه‌ی کنترل پیش بین به همراه محصول آن‌ها، راهکاری که برای حل مسئله بهینه سازی در کنترل پیش بین مورد استفاده قرار داده اند شرح داده می‌شود.

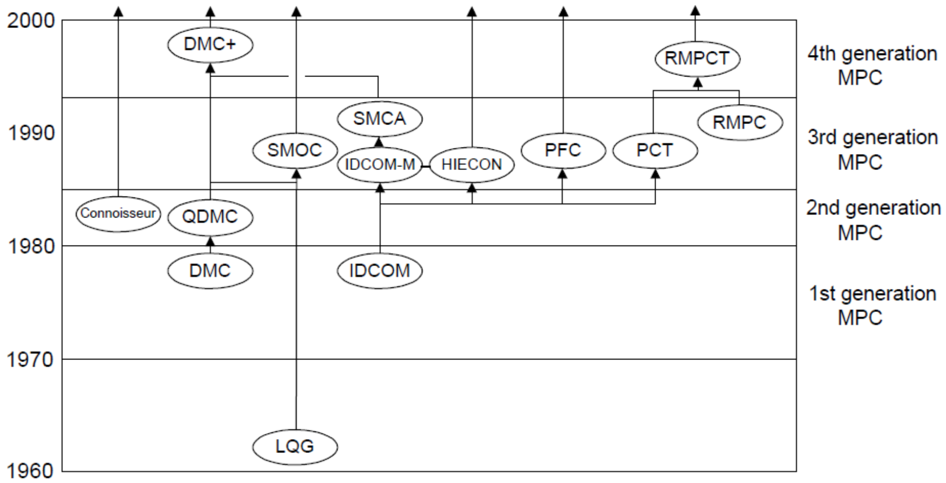
همان طور که در شکل زیر مشخص است، کاربرد اولین گونه های کنترل پیش بین به مسئله LQG باز می‌گردد. اگرچه توسعه و کاربرد کنترل پیش بین در صنعت صورت گرفت، ایده‌ی کنترل یک سیستم با حل یک دنباله مسائل بهینه سازی پویای حلقه باز، پیش از آن هم ارائه شده بود[۳].

در اینجا از ذکر تاریخچه کنترل پیش بین و ترتیبی که توسعه یافتند، پرهیز می‌شود و تنها به مقایسه آن‌ها پرداخته می‌شود. در




شکل۱-۱بسته‌های نرم افزاری تجاری ارائه شده برای کنترل پیش بین و زمان ارائه آن‌ها نشان داده شده است.

[1]Salahi-Terlaki
[2]MATLAB




**شکل۱-۱: بسته‌های نرم افزاری تجاری ارائه شده در کنترل پیش بین به همراه زمان ارائه**

شرح کنترل کننده های پیش بین خطی موجود در صنعت در جدول۱-۱ مشخص شده است.

**جدول۱-۱ :مقایسه‌ی کنترل کننده های صنعتی پیش بین موجود**

| Company | Product Name | Description |
|---|---|---|
| Adersa | HIECON | Hierarchical constraint control |
| | PFC | Predictive functional control |
| | GLIDE | Identification package |
| Aspen Tech | DMC-plus | Dynamic matrix control package |
| | DMC-plus model | Identification package |
| Honeywell Hi-Spec | RMPCT | Robust model predictive control technology |
| Shell Global Solutions | SMOC-II | Shell multivariable optimizing control |
| Invensys | Connoisseur | Control and identification package |

معمولاً در این کنترل کننده‌ها مسئله بهینه سازی طوری فرموله می‌شود که ورودی و خروجی‌های حالت ماندگار را تا جایی که ممکن است، به اهداف تعیین شده نزدیک کند، بدون آن‌که قیود خروجی و ورودی سیستم را نقض کند.

کنترل کننده شرکت اینونسیس[1] برای انجام بهینه سازی در حالت ماندگار از یک مدل برنامه ریزی خطی استفاده می‌کند. مشخصه متمایز کننده برنامه ریزی خطی این است که اهداف بهینه بر روی رئوس قیود واقع می‌شوند. اگر مرز قیدها دائماً به خاطر نویز تغییر کند، پاسخ بهینه ممکن است نوسان کند که

[1]Invensys

باعث کاهش شدید کارایی کلی کنترل کننده می‌شود. عموماً راه حل این مسئله فیلتر کردن شدید سیگنال خروجی است.

کنترل کننده های RMPCT,PFC, Aspen Target, MVC & Process perfecter از برنامه ریزی درجه دوم برای محاسبه هدف استفاده می‌کنند. پاسخ مسئله برنامه ریزی درجه دوم الزاماً بر روی مرز قیدها واقع نمی‌شود، بنابراین پاسخ‌ها به اندازه برنامه ریزی خطی نوسان نخواهند داشت.

همان طور که گفته شد، اکثر کنترل کننده های صنعتی از برنامه ریزی درجه دوم برای مدل سازی مسئله بهینه سازی کنترل پیش بین استفاده می‌کنند. این مسئله برنامه ریزی درجه دوم را می‌توان با بسته‌های نرم افزاری تجاری بهینه سازی حل کرد، اما برای مسائل بسیار بزرگ، یا برای فرآیندهای بسیار سریع، زمان کافی برای حل مسئله برنامه ریزی درجه دوم وجود ندارد، به همین دلیل الگوریتم‌های DMC-plus & PFC از الگوریتم‌های سریع زیربهینه[1] برای حل تقریبی مسئله استفاده می‌کنند. الگوریتم PFC محاسبات را بدون قید انجام می‌دهد. سپس اگر مقادیر ورودی، قیود سخت[2] را نقض کرده بودند، آن ورودی‌ها را حذف می‌کند. این کار به طور کلی کارایی را کاهش می‌دهد. کارایی عموماً در سطح قابل قبولی است، اما پاسخ حاصل شرایط بهینگی کاروش-کیون-تاکر[3] را نقض می‌کند.

در جدول۱-۲ مقایسه‌ی مناسبی بین کنترل کننده های ارائه شده صنعتی انجام شده است.

**جدول۱-۲: مقایسه بین کنترل کننده های خطی موجود صنعتی**

| SGS | Invensys | Adersa | Adersa | Honeywell Hi-Spec | Aspen Tech | شرکت سازنده |
|---|---|---|---|---|---|---|
| SMOC | Connois. | PFC | HIECON | RMPCT | DMC-plus | محصول |
| LSS | ARX, FIR | LSS, TF, ARX | FIR | ARX, TF | FSR | مدل خطی |
| Q[I,O],R | L[I,O] | Q[I,O] | - | Q[I.O] | L/Q[I,O],...,R | معیار بهینه سازی حالت ماندگار |
| ASQP | ASQP | LS | ASQP | ASQP | SLS | روش حل |

در جدول بالا حروف نگاشته شده مخفف مفاهیم زیر هستند:

**جدول۱-۳: معنی حروف مخفف جدول۱-۲**

| معنی | حروف اختصاری |
|---|---|
| Quadratic | Q |
| Inputs | I |
| Outputs | O |

[1]Sub-optimal
[2]Hard constraint
[3]Karush-Kuhn-Tucker

| | |
|---|---|
| M | Input moves |
| S | Sub-optimal solution |
| LS | Least squares |
| SLS | Sequential LS |
| ASQP | Active set quadratic program |

همچنین برخی از شرکت‌ها نیز از روش‌های نقطه درونی برای بهینه سازی استفاده کرده‌اند (برای مثال به [۴۲] مراجعه شود).

## ۱-۶- نوآوری پژوهش

در این پایان نامه، الگوریتم ارائه شده توسط صلاحی (الگوریتم اصلاح شده‌ی مهروترا) به مسئله‌ی کنترل پیش بین تعمیم یافته، اعمال شده است. برای این کار، با توجه به این که این الگوریتم در ابتدا برای مسئله برنامه ریزی خطی توسعه یافته بود، الگوریتم برای مسئله‌ی برنامه ریزی درجه دوم که در اکثر حالات کنترل پیش بین با آن روبرو هستیم، توسعه یافته است. سپس برای هر دو الگوریتم پیش‌گو-اصلاح‌گر مهروترا و اصلاح شده‌ی مهروترا برنامه‌ی متلب نوشته شده است و این برنامه‌ها برای چندین مسئله‌ی کنترل پیش بین امتحان شده‌اند و نتایج شبیه سازی با نتایج مشابه که از اعمال دستور بهینه سازی درجه دوم جعبه ابزار بهینه سازی متلب (نگارش ۵,۰) به مسئله‌ی کنترل پیش بین حاصل شده است، از لحاظ سرعت پاسخ‌دهی مقایسه شده‌اند.

## ۱-۷- ساختار پایان نامه

در این پایان نامه، در فصل دوم مسئله کنترل بهینه به صورت مختصر مطرح می‌شود. پس از توضیحات کلی و نقش مسئله بهینه سازی در کنترل پیش بین، راه حل‌های سرعت بخشیدن به حل مسئله مطرح می‌شود. در فصل سوم تعریف کلی مسئله بهینه سازی درجه دوم و خطی به همراه الگوریتم‌های پیش‌گو-اصلاح‌گر مهروترا و مهروترای اصلاح شده همراه با مقدمات و پیش‌نیازهای ریاضی آن مطرح می‌شود و الگوریتم مهروترای اصلاح شده برای مسئله برنامه ریزی درجه دوم توسعه داده می‌شود. سپس در فصل چهارم نتایج شبیه سازی چندین مسئله کنترل پیش بین ارائه می‌شود و سرعت رسیدن به پاسخ توسط دستور نرم افزار متلب و برنامه‌ی نوشته شده برای الگوریتم مهروترای اصلاح شده، با هم مقایسه می‌شوند.

# فصل ۲- کنترل پیش بین

## ۲-۱- پیش گفتار

در این فصل ابتدا به معرفی کنترل پیش بین و مزیت‌ها و معایب آن می‌پردازیم، سپس الگوریتم کنترل پیش بین تعمیم یافته GPC[1]را به طور دقیق‌تر مورد بررسی قرار خواهیم داد و معادلات آن شرح داده خواهد شد. پس از آن به نقش بهینه سازی در مسئله‌ی کنترل پیش بین و مدل‌های متفاوتی که برای آن در مسئله کنترل پیش بین ارائه شده است می‌پردازیم و از آنجا که در نهایت هدف از ارائه این پایان نامه سرعت بخشیدن به حل مسئله کنترل پیش بین است، راه های سرعت بخشیدن به مسئله کنترل پیش بین ارائه خواهد شد.

## ۲-۲- معرفی کنترل کننده های پیش بین

کنترل پیش بین یکی از الگوریتم‌های کنترلی است که طی سالیان اخیر رشد چشمگیری داشته است. عبارت کنترل پیش بین مشخص کننده‌ی یک راهکار یکتای کنترلی نیست بلکه مشخص کننده‌ی بازه‌ی وسیعی از روش‌های کنترلی است که از مدل فرآیند برای محاسبه‌ی سیگنال کنترلی از طریق کمینه کردن یک تابع معیار استفاده می‌کنند. تفاوت الگوریتم‌های متفاوت کنترل پیش بین در مدلی که از آن برای توصیف فرآیند و نویز استفاده می‌کنند و نیز در تابع معیار آن‌ها است.

به طور کلی مزیت‌های کنترل پیش بین عبارتند از[۴۳]:

- کنترل پیش بین برای کاربرانی با دانش محدود از کنترل جذاب است زیرا مفاهیم آن بسیار حسی هستند و تنظیم آن نسبتاً آسان است.

- این کنترل کننده‌ها را می‌توان برای بازه‌ی وسیعی از کاربردها استفاده کرد، از سیستم‌های ساده گرفته تا سیستم‌های پیچیده و حتی سیستم‌های نامینیمم فاز و با تأخیر خیلی زیاد.

- کار کردن با کنترل پیش بین چند متغیره بسیار آسان است.

- به طور ذاتی برای زمان مرده سیستم جبران سازی انجام می‌دهد.

[1]Generalized Predictive Control

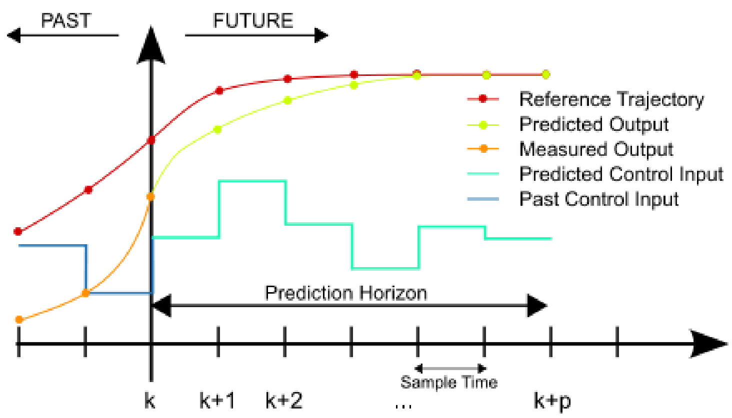
- قیود سیستم به صورت صریح در آن ذکر می‌شوند و بنابراین هم قیود سخت و هم قیود نرم[۱] به آسانی مدنظر قرار می‌گیرند.

البته کنترل پیش بین معایب خاص خود را نیز دارد، از جمله عیوب کنترل پیش بین نیاز کنترل پیش بین به دانستن یک مدل از سیستم است. بنابراین برای بکار بردن کنترل پیش بین نیاز به داشتن دانش پیشین از سیستم داریم، که این همیشه مقدور نیست.

مشکل دیگر کنترل پیش بین این است که اگر چه هنگامی که دینامیک سیستم تغییر نمی‌کند می‌توان سیگنال کنترلی را از پیش تعیین کرد، اما در حالت تطبیقی که دینامیک سیستم متغیر است در هر زمان نمونه برداری محاسبات باید یک بار به طول کامل انجام شود. به عبارتی مسئله‌ی بهینه سازی یکبار حل می‌گردد که حل این مسئله‌ی بهینه سازی نیز بسیار زمان‌بر است. اگرچه امروزه با پیشرفت رایانه‌ها مشکل محاسبات رایانه ای کمی مرتفع شده است اما باید در ذهن داشت که زمانی که رایانه در دسترس دارد در واقع باید برای انجام اعمال دیگری مثل ارتباطات، ثبت داده‌ها، اخطار و ... استفاده شود. اما با این حال کنترل پیش بین به خوبی در صنعت رایج شده است و از آن استفاده می‌شود.

## ۲-۲-۱- راهکار کنترل پیش بین

راهکار تمامی کنترل کننده های پیش بین به صورت شکل۲-۱ است:




**شکل۲-۱: راهکار کلی کنترل کننده های پیش بین**

در هر لحظه $t$ با استفاده از مدل فرآیند، خروجی‌های آینده‌ی سیستم برای یک افق معین $N$ که به آن افق پیش بینی[۲] گفته می‌شود، تعیین می‌شوند. این خروجی‌ها $y(t+k|t)$ برای $k = 1 \dots N$ به

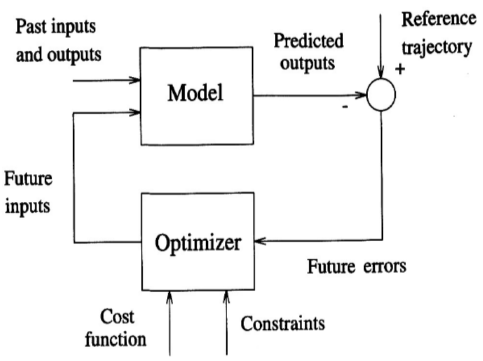
[1]Soft Constraints
[2]Prediction Horizon

مقادیر ورودی و خروجی پیشین (تا لحظه $t$) و به سیگنال‌های کنترل در لحظات آینده $u(t+k|t), k = 0 \dots N-1$ بستگی دارند که این مقادیر سیگنال‌های کنترلی در لحظات آینده باید محاسبه شوند.

در این مرحله، سیگنال‌های کنترلی در لحظات آینده با بهینه کردن یک ضابطه (تابع معیار) بدست می‌آیند که تلاش در این تابع معیار آن است که فرآیند تا جای ممکن به مسیر مرجع[1] نزدیک گردد. این ضابطه عموماً به شکلِ تابعی درجه دوم از خطای بینِ سیگنالِ پیش بینی شده‌ی خروجی و مسیرِ مرجع است. در اکثر حالات، تلاش کنترلی[2] در تابع معیار لحاظ می‌شود. اگر مسئله نامقید باشد، مقدار سیگنال کنترلی را می‌توان به صورت صریح بدست آورد اما اگر قید در مسئله داشته باشیم باید مسئله‌ی بهینه سازی به صورت عددی با الگوریتم‌های خاص بهینه سازی حل شود.

حال سیگنال کنترلی محاسبه شده $u(t|t)$ به فرآیند اعمال می‌شود، اما بقیه‌ی سیگنال‌های کنترلی محاسبه شده استفاده نمی‌شوند زیرا در گام بعدی مقدار $y(t+1)$ را می‌دانیم و باید مجدداً محاسبات انجام شوند. بنابراین در گام جدید $u(t+1|t+1)$ محاسبه می‌شود که با مقدار $u(t+1|t)$ که در گام قبل محاسبه کردیم متفاوت خواهد بود.

برای فهم بهتر از نحوه ساختار کنترل کننده های پیش بین شکل۲-۲ را ببینید.




**شکل۲-۲: ساختار کنترل کننده های پیش بین (برگرفته از مرجع[۴۳])**

## ۲-۲-۲- انواع کنترل کننده های پیش بین

اگرچه همه‌ی کنترل کننده های پیش بین مؤلفه های مشترکی دارند، اما در نکات زیر با هم تفاوت دارند:

[1] Reference Trajectory
[2] Control Effort

۱- مدل سیستم
۲- تابع معیار
۳- نحوه‌ی بدست آوردن قانون کنترل

**مدل سیستم:**

**پاسخ ضربه:**

برخی کنترل کننده های پیش بین مثل $MAC$ که حال خاصی از $GPC$ و $EPSAC$ است از این مدل استفاده می‌کنند. رابطه‌ی خروجی و ورودی به شکل زیر است:

$$u(t) = \Sigma_{i=1}^{\infty} h_i u(t-i)$$

$$y(t) = \Sigma_{i=1}^{N} h_i u(t-i) = H(z^{-1})u(t)$$

$$\hat{y}(t+k|t) = \Sigma_{i=1}^{N} h_i u(t+k-i|t) = H(z^{-1})u(t+k|t) \qquad (۱-۲)$$

**پاسخ پله:**

این روش توسط الگوریتم‌های $DMC$ و الگوریتم‌های اصلاح شده‌ی $DMC$ به کار گرفته می‌شود. برای سیستم‌های پایدار، پاسخ به صورت زیر است:

$$y(t) = y_0 + \Sigma_{i=0}^{N} g_i \Delta u(t-i) = y_0 + G(z^{-1})(1-z^{-1})u(t) \qquad (۲-۲)$$

که در آن $g_i$ مقادیر نمونه برداری شده‌ی خروجی برای ورودی پله است و $\Delta u(t) = u(t) - u(t-1)$ همچنین:

$$\hat{y}(t+k|t) = \Sigma_{i=1}^{N} g_i \Delta u(t+k-i|t) \qquad (۳-۲)$$

**تابع تبدیل:**

این روش توسط الگوریتم‌های $GPC, UPC, EPSAC, EHAC, MUSMAR$ استفاده می‌شود. در این حالت خروجی به صورت زیر داده می‌شود:

$$A(z^{-1})y(t) = B(z^{-1})u(t)$$

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{na} z^{-na}$$

$$B(z^{-1}) = b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{nb} z^{-nb}$$

$$\hat{y}(t+k|t) = \frac{B(z^{-1})}{A(z^{-1})} u(t+k|t) \qquad (۴-۲)$$

**مدل فضای حالت:**

این مدل توسط الگوریتم $PFC$ استفاده می‌شود و به صورت زیر است:

$$x(t) = Ax(t-1) + Bu(t-1)$$

$$y(t) = Cx(t)$$

$$\hat{y}(t+k|t) = C\hat{x}(t+k|t) = C[A^k x(t) + \Sigma_{i=1}^{k} A^{i-1} Bu(t+k-i|t)] \qquad (۵-۲)$$

در این پایان نامه از کنترل پیش بین تعمیم یافته $(GPC)$ استفاده شده است. در ادامه توصیفات و روابط کنترل پیش بین تعمیم یافته ارائه خواهد شد.

## ۲–۳– کنترل پیش بین تعمیم یافته $(GPC)$

الگوریتم $GPC$ توسط کلارک[۴۴] ارائه شد و یکی از محبوب‌ترین الگوریتم‌های کنترل پیش بین هم در صنعت و هم در جوامع آکادمیک بوده است. [۴۳] ایده‌ی اصلی کنترل پیش بین تعمیم یافته محاسبه‌ی سیگنال‌های کنترلی آینده به صورتی است که یک تابع هزینه‌ی چند مرحله ای را در یک افق پیش بینی مشخص کمینه کند. این کنترل کننده زمانی که نامقید است می‌تواند سیگنال کنترلی را به صورت تحلیلی ارائه کند، همچنین می‌تواند در مورد سیستم‌های ناپایدار و نامینیمم فاز به کار برده شود.
معادلات کنترل پیش بین تعمیم یافته به صورت زیر است:
برای یک سیستم تک ورودی- تک خروجی داریم:

$$A(z^{-1})y(t) = z^{-d} B(z^{-1}) u(t-1) + C(z^{-1}) e(t) \qquad (۶-۲)$$

که در آن $u(t)$ دنباله‌ی ورودی‌ها و $y(t)$ دنباله‌ی خروجی‌ها است و $e(t)$ نویز سفید با میانگین صفر است. داریم:

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{na} z^{-na} \qquad (۷-۲)$$

$$B(z^{-1}) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{nb} z^{-nb} \quad (۲-۸)$$

$$C(z^{-1}) = 1 + c_1 z^{-1} + c_2 z^{-2} + \cdots + c_{nc} z^{-nc} \quad (۲-۹)$$

که در روابط بالا $d$ زمان مرده‌ی[1] سیستم است. این مدل با عنوان مدل کنترل کننده با میانگین متحرک رگرسیو خودکار $(CARMA)$[2] شناخته می‌شود. برای بسیاری از صنایع مدل $CARIMA$[3] مناسب‌تر است. این مدل به صورت زیر است:

$$A(z^{-1})y(t) = B(z^{-1})z^{-d}u(t-1) + C(z^{-1})\frac{e(t)}{\Delta} \quad ,\Delta = 1 - z^{-1} \quad (۲-۱۰)$$

الگوریتم کنترل پیش بین تعمیم یافته شامل اِعمال کردن دنباله‌ی سیگنال کنترلی است که یک تابع معیار به فرم زیر را کمینه کند:

$$J(N_1, N_2, N_u) = \Sigma_{j=N_1}^{N_2} \delta(j)[\hat{y}(t+j|t) - w(t+j)]^2 + \Sigma_{j=1}^{N_u} \lambda(j)[\Delta u(t+j-1)]^2 \quad (۲-۱۱)$$

در عبارت بالا $\hat{y}(t+j|t)$ خروجی بهینه‌ی پیش بینی شده برای سیستم در $j$ گام جلوتر است که بر اساس اطلاعات تا لحظه‌ی $t$ بدست می‌آید. متغیرهای $N_1$ و $N_2$ ابتدا و انتهای افقی که هزینه را برای آن تعریف می‌کنیم است و $N_u$ افق کنترلی است. متغیرهای $\delta(j)$ و $\lambda(j)$ دنباله های وزنی و $w(t+j)$ منحنی مرجع آینده هستند.

هدف کنترل پیش بین این است که دنباله‌ی کنترلی آینده ... ,$u(t), u(t+1)$ را طوری بدست آوریم که خروجی‌های آینده‌ی فرآیند $y(t+j)$ نزدیک به $w(t+j)$ شوند. این کار با کمینه کردن $J(N_1, N_2, N_u)$ انجام می‌شود. معادله‌ی دیوفانتین[4] زیر را در نظر بگیرید:

$$1 = E_j(z^{-1})\tilde{A}(z^{-1}) + z^{-j}F_j(z^{-1}), \quad \tilde{A}(z^{-1}) = \Delta A(z^{-1}) \quad (۲-۱۲)$$

چندجمله‌ای های $E_j$ و $F_j$ به صورت یکتا تعریف می‌شوند. این چندجمله‌ای‌ها را می‌توان از تقسیم کردن ۱ بر $\tilde{A}(z^{-1})$ تا زمانی که باقیمانده را بتوان به صورت $z^{-j}F_j(z^{-1})$ نوشت بدست آورد. خارج قسمت تقسیم عبارت $E_j(z^{-1})$ است. حال معادله‌ی (۲-۱۰) را تقسیم بر $\Delta E_j(z^{-1})z^j$ می‌کنیم:

---

[1] Dead time

[2] Controller Auto-Regressive Moving-Average

[3] Controller Auto-Regressive Integrated Moving-Average

[4] Diophantine equation

$$\tilde{A}(z^{-1})E_j(z^{-1})y(t+j) = E_j(z^{-1})B(z^{-1})\Delta u(t+j-d-1) + E_j(z^{-1})e(t+j) \quad (۱۳\text{-}۲)$$

این معادله را می‌توان به صورت زیر نوشت:

$$\left(1 - z^{-j}F_j(z^{-1})\right)y(t+j) = E_j(z^{-1})B(z^{-1})\Delta u(t+j-d-1) + E_j(z^{-1})e(t+j)$$

$$y(t+j) = F_j(z^{-1})y(t) + E_j(z^{-1})B(z^{-1})\Delta u(t+j-d-1) + E_j(z^{-1})e(t+j) \quad (۱۴\text{-}۲)$$

از آنجا که درجه‌ی چند جمله ای $E_j(z^{-1}) = j-1$ است، جملات مربوط به نویز در معادله‌ی بالا همگی مربوط به آینده هستند. بنابراین بهترین پیش بینی برای $y(t+j)$ به صورت زیر است:

$$\hat{y}(t+j|t) = G_j(z^{-1})\Delta u(t+j-d-1) + F_j(z^{-1})y(t) \quad (۱۵\text{-}۲)$$

که در آن $G_j(z^{-1}) = E_j(z^{-1})B(z^{-1})$

حال فرض کنید که چندجمله‌ای های زیر بدست آمده‌اند:

$$F_j(z^{-1}) = f_{j,0} + f_{j,1}z^{-1} + \cdots + f_{j,na}z^{-na} \quad (۱۶\text{-}۲)$$

$$E_j(z^{-1}) = e_{j,0} + e_{j,1}z^{-1} + \cdots + e_{j,j-1}z^{-(j-1)} \quad (۱۷\text{-}۲)$$

فرض کنید که چندجمله‌ای زیر را نیز بدست آورده‌ایم:

$$F_{j+1}(z^{-1}) = f_{j+1,0} + f_{j+1,1}z^{-1} + \cdots + f_{j+1,na}z^{-na} \quad (۱۸\text{-}۲)$$

می‌توان دید که:

$$E_{j+1}(z^{-1}) = E_j(z^{-1}) + e_{j+1,j}z^{-j} \quad (۱۹\text{-}۲)$$

که در آن $e_{j+1,j} = f_{j,0}$ .

ضرایب چندجمله‌ای $F_{j+1}$ را می‌توان به صورت زیر نوشت:

$$f_{j+1,i} = f_{j,i+1} - f_{j,0}\bar{a}_{i+1} \quad i = 0 \ldots na-1 \quad (۲۰\text{-}۲)$$

چندجمله‌ای $G_{j+1}$ را می‌توان به صورت بازگشتی از عبارت زیر بدست آورد:

$$G_{j+1} = E_{j+1}B = \left(E_j + f_{j,0}z^{-j}\right)B$$

$$G_{j+1} = G_j + f_{j,0}z^{-j}B$$

$$g_{j+1,j+1} = g_{i,j+i} + f_{j,0}b_i \qquad (۲۱\text{-}۲)$$

برای حل مسئله کنترل پیش بین تعمیم یافته، باید از بهینه کردن تابع هزینه‌ای که پیش از این مطرح شد مجموعه‌ی سیگنال‌های کنترلی $u(t), u(t+1), \ldots, u(t+N)$ بدست آیند. حال فرض کنید که مقادیر دنباله های زیر را برای پیش بینی‌های بهینه در $j$ گام جلوتر داریم:

$$\hat{y}(t+d+1|t) = G_{d+1}\Delta u(t) + F_{d+1}y(t)$$

$$\hat{y}(t+d+2|t) = G_{d+2}\Delta u(t+1) + F_{d+2}y(t)$$

$$\vdots$$

$$\hat{y}(t+d+N|t) = G_{d+N}\Delta u(t+N-1) + F_{d+N}y(t) \qquad (۲۲\text{-}۲)$$

که می‌توان آن را به صورت زیر نوشت:

$$y = Gu + F(z^{-1})y(t) + G'(z^{-1})\Delta u(t-1)$$

$$y = \begin{bmatrix} \hat{y}(t+d+1|t) \\ \hat{y}(t+d+2|t) \\ \vdots \\ \hat{y}(t+d+N|t) \end{bmatrix} \quad u = \begin{bmatrix} \Delta u(t) \\ \Delta u(t+1) \\ \vdots \\ \Delta u(t+N-1) \end{bmatrix} \quad G = \begin{bmatrix} g_0 & 0 & \ldots & 0 \\ g_1 & g_0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ g_{N-1} & g_{N-2} & \ldots & g_0 \end{bmatrix}$$

$$G'(z^{-1}) = \begin{bmatrix} (G_{d+1}(z^{-1}) - g_0)z \\ (G_{d+2}(z^{-1}) - g_0 - g_1z^{-1})z^2 \\ \vdots \\ (G_{d+N}(z^{-1}) - g_0 - g_1z^{-1} - \cdots - g_{N-1}z^{-(N-1)}z^N \end{bmatrix}$$

$$F(z^{-1}) = \begin{bmatrix} F_{d+1}(z^{-1}) \\ F_{d+2}(z^{-1}) \\ \vdots \\ F_{d+N}(z^{-1}) \end{bmatrix} \qquad (۲۳\text{-}۲)$$

دقت کنید که دو جمله‌ی آخر تنها به زمان‌های گذشته وابسته هستند و می‌توان چندجمله‌ای را به صورت زیر نوشت:

$$y = Gu + f \qquad (۲۴\text{-}۲)$$

اگر تمامی شرایط اولیه سیستم صفر باشند مقدار $f$ نیز صفر است.

اکنون با این مقادیر جدید تابع هزینه (۲-۱۱) را می‌توان به صورت زیر نوشت:

$$J = (Gu + f - w)^T (Gu + f - w) + \lambda u^T u \qquad (۲\text{-}۲۵)$$

که در آن

$$w = [w(t+d+1) \quad w(t+d+2) \quad \dots \quad w(t+d+N)]^T \qquad (۲\text{-}۲۶)$$

معادله‌ی بالا را می‌توان به صورت زیر نوشت:

$$J = \frac{1}{2} u^T H u + b^T u + f_0 \qquad (۲\text{-}۲۷)$$

که در آن

$$H = 2(G^T G + \lambda I)$$

$$b^T = 2(f - w)^T G$$

$$f_0 = (f - w)^T (f - w) \qquad (۲\text{-}۲۸)$$

همان طور که در معادله‌ی (۲-۲۷) مشاهده می‌شود در نهایت به یک مسئله‌ی برنامه ریزی درجه دوم رسیده‌ایم. با دقت به این معادلات می‌توان مشاهده کرد که مقدار $f_0$ اگر چه در مقدار مطلق $J$ نقش دارد اما در مقدار $u$ ای که به ازای آن تابع هزینه بهینه است تأثیری ندارد بنابراین می‌توان در حل مسئله بهینه سازی درجه دوم این پارامتر را برای بدست آوردن سیگنال کنترلی حذف کرد. اکنون این مسئله برنامه ریزی درجه دوم باید حل شود، در منابع و مراجع مختلف برای حل مسائل بهینه سازی درجه دوم روش‌های گوناگونی پیشنهاد گردیده است که بررسی مفصل آن‌ها را به فصل سوم واگذار می‌کنیم.

## ۲–۴– نقش مسئله‌ی بهینه سازی در کنترل پیش بین

همان طور که مشاهده می‌شود اِعمال سیگنال کنترلی مناسب در گرو محاسبه‌ی درست آن از طریق حل مسئله‌ی بهینه سازی است. اما دقت کنید که در بهینه سازی عددی، بسیاری از الگوریتم‌ها برای مسئله دارای جواب نهایی نیستند، به این معنی که هیچ‌وقت به پاسخ نهایی به طور دقیق نمی‌رسند بلکه به سمت آن میل می‌کنند و با هر بار انجام دوباره‌ی الگوریتم‌های تکرار شونده‌ی بهینه سازی، تنها در همسایگی کوچک‌تری از جواب حقیقی مسئله قرار می‌گیرند. بنابراین برای خروج الگوریتم از تکرار در این گونه الگوریتم‌ها یک مقدار خطا در نظر می‌گیرند که هر گاه پاسخ در آن محدوده‌ی معین واقع شد،

الگوریتم از تکرار خارج می‌شود. (از جمله‌ی این الگوریتم‌ها الگوریتم پیش‌گو-اصلاح‌گر مهروترا است که بر خلاف سیمپلکس، هرگز به جواب نهایی نمی‌رسد[۴۵]) اکنون می‌توان به نقش اساسی مسئله‌ی بهینه سازی در کنترل پیش بین پی برد. اگر این مقدار خطا که شرط خروج الگوریتم بهینه سازی از تکرار است بسیار کوچک انتخاب شود، اگرچه الگوریتم بهینه سازی در نهایت به میزان خطای تعیین شده به جواب نهایی نزدیک می‌شود، اما از آنجا که دقت زیادی مدنظر بوده است تعداد تکرارهای الگوریتم تکرار شونده‌ی بهینه سازی افزایش می‌یابد و بنابراین مدت زمانی که طول می‌کشد تا به پاسخ برسد خیلی زیاد می‌شود. از طرفی اگر مقدار خطایی که شرط خروج از تکرارها در مسئله‌ی بهینه سازی است بسیار بزرگ باشد، جواب مسئله‌ی بهینه سازی (که در کنترل پیش بین تعمیم یافته همان سیگنال کنترل ورودی به سیستم است) دارای دقت لازم نیست و بنابراین کارایی[۱] سیستم را کاهش می‌دهد و در بسیاری از موارد حتی منجر به ناپایداری سیستم می‌شود (برای اطلاعات بیشتر به بخش پیشنهادات مراجعه گردد).
اکنون ببینیم که در کنترل پیش بین، چند نوع مسئله‌ی بهینه سازی تعریف می‌شود.

## ۲-۵- مدل سازی مسئله‌ی کنترل پیش بین به صورت مسئله‌ی بهینه سازی

برای آن‌که به طور ویژه بر روی بهینه سازی در کنترل پیش بین دقیق شویم احتیاج است که مسئله‌ی کنترل پیش بین را به صورت یک مسئله‌ی بهینه سازی مدل کنیم، در این راستا تلاش‌های بسیاری انجام شده است و مدل‌های مختلفی ارائه شده است که در زیر برخی از آن‌ها به اختصار ذکر خواهد شد.[۴،۴۶]

اکثر کارهایی که در حوزه‌ی کنترل پیش بین انجام شده‌اند، از مدل‌های خطی و معیارهای درجه دوم برای عملکرد استفاده کرده‌اند. مدل‌های رایج، مدل‌های پاسخ پله و ضربه، مدل‌های ARMAX/CARIMA و مدل فضای حالت هستند.

در معیارهای عملکرد (توابع هزینه)، عموماً برای انحراف کنترل[۲]، متغیر کنترل و/یا تغییرات سیگنال کنترلی در هر گام[۳]، وزن‌های درجه دوم در نظر می‌گیرند. در عمل ورودی کنترلی باید در میان یک حد بالایی و پایینی محصور و مقید باشد. ضمناً گاهی مقید کردن متغیرهای پاسخ هم مطلوب است به این معنی که خروجی هم باید در یک محدوده‌ی مشخص قرار داشته باشد. بنابراین اکثر شیوه های فرمول‌بندی کنترل پیش بین از مدل برنامه ریزی درجه دوم استفاده می‌کنند.

[1]Performance
[2]Control deviation
[3]Control increment

فرموله کردن الگوریتم‌های کنترل پیش بین عموماً بسیار طولانی است و پس از بالا و پایین کردن و تغییرات بسیار منجر به یک مسئله برنامه ریزی درجه دوم می‌شود که در آن، مجهولات ورودی‌های کنترلی آینده هستند. یک رویکرد جایگزین، این است که متغیرهایی مثل حالت‌ها، خروجی‌ها، انحرافات کنترلی و غیره را مجهول در نظر بگیریم و مدل را به صورت قیدهای خطی مساوی در مسئله‌ی برنامه ریزی درجه دوم در نظر بگیریم.

**مسئله‌ی کنترل پیش بین به صورت فضای حالت**:

مدل فضای حالت زیر را فرض کنید:

$$x_{k+1} = Ax_k + Bu_k \tag{۲-۲۹}$$

$$y_k = Cx_k + Du_k \tag{۲-۳۰}$$

که در آن $u_k \in \mathbb{R}^{m\times 1}$ ورودی کنترلی است و $y_k \in \mathbb{R}^{r\times 1}$ خروجی کنترل شده است و $x_k \in \mathbb{R}^{n\times 1}$ حالت‌ها است، و معیار عملکرد $J_k$ به صورت زیر تعریف می‌شود:

$$J_k = \sum_{i=0}^{\infty}(e_{k+i}^T Q e_{k+i} + u_{k+i}^T R u_{k+i} + \Delta u_{k+i}^T S \Delta u_{k+i}) \tag{۲-۳۱}$$

در معادلات بالا داریم:

$$e_k = y_k - s_k \tag{۲-۳۲}$$

که $s_k$ ورودی مرجع است، و تغییرات کنترلی $\Delta u_k$ به صورت زیر تعریف می‌شود:

$$\Delta u_k = u_k - u_{k-1} \tag{۲-۳۳}$$

برای سیستم‌های با حلقه‌ی باز پایدار، با اعمال چند شرط نه چندان سخت‌گیرانه، می‌توانیم معیار عملکرد نامحدود بالا را به معیار زیر با عملکرد محدود تبدیل کنیم (برای توضیحات بیشتر به [۴۷] مراجعه گردد).

$$\begin{aligned} J_k = {} & x_{k+N}^T \bar{Q} x_{k+N} + u_{k+N-1}^T S u_{k+N-1} \\ & + \sum_{i=0}^{N-1}(e_{k+i}^T Q e_{k+i} + u_{k+i}^T R u_{k+i} + \Delta u_{k+i}^T S \Delta u_{k+i}) \end{aligned} \tag{۲-۳۴}$$

که در اینجا $\bar{Q}$ را می‌توان با حل معادله‌ی لیاپانف گسسته در زمان زیر بدست آورد:

$$\bar{Q} = C^T Q C + A^T \bar{Q} A \qquad (۲-۳۵)$$

وقتی که $x_k$ معین است و دنباله‌ی ورودی‌های بهینه را با $u_{(k+i|k)}$ و $i \in \{0, \dots, N-1\}$ نمایش می‌دهیم، با حل تکرار شونده‌ی مسئله‌ی کنترل بهینه برای هر بار تکرار $k$ و با فرض ورودی کنترلی به صورت $u_k = u_{k|k}$ به یک سیستمِ حلقه بسته‌ی پایدار نامی[1] می‌رسیم (برای اطلاعات بیشتربه[۴۸] مراجعه شود).

یکی از مزیت‌های اصلی کنترل پیش بین برخورد صریح و مستقیم آن با قیدها در محاسبه‌ی ورودی کنترلی است. این از لحاظ صنعتی بسیار مهم است زیرا تمامی فرآیندها محدودیت و قید دارند، همچنین عملگرها[2] نیز یک بازه‌ی عملکرد دارند:

$$u^l \leq u \leq u^u \qquad (۲-۳۶)$$

و نیز تغییرات کنترلی محدودی دارند:

$$\Delta u^l \leq \Delta u_k \leq \Delta u^u \qquad (۲-۳۷)$$

قیدهای خروجی عموماً به دلیل مسائل ایمنی و بهبود عملکرد سیستم در نظر گرفته می‌شوند. همچنین این قیود زمانی که مقدار دقیق بعضی از خروجی‌ها از اهمیت کمتری برخوردار است و تنها کافی است که در یک محدوده‌ی خاص باشد در نظر گرفته می‌شوند. این قیود به صورت زیر بیان می‌گردند:

$$y^l \leq y \leq y^u \qquad (۲-۳۸)$$

## ۲–۵–۱– انواع شیوه‌های فرمول بندی مسئله‌ی کنترلِ پیش بین به صورت برنامه ریزی درجه دوم

### ۲-۵-۱-۱- مسئله‌ی برنامه ریزی درجه دوم استاندارد

مسئله‌ی کنترل پیش بین در حالت کلی، که در بالا گفته شد را می‌توان به صورت زیر به یک مسئله‌ی برنامه ریزی درجه دوم تبدیل کرد:

[1] Nominally stable closed loop system
[2] Actuators

$$\min_z F(z) = \frac{1}{2} z^T H z + g^T z + k$$

$$s.t. \quad Az = a$$

$$Bz \le b$$

$$z^l \le z \le z^u \tag{۲-۳۹}$$

از آنجایی که مقدار متغیّر $k$ در تعیینِ مقدارِ بهینه‌ی $z$ تأثیری ندارد، در نظر گرفته نمی‌شود. اگر قیود نامساوی فعال نباشند (هنگامی که قیود نامساوی در حالتِ مرزی محقق شوند و به قیودِ مساوی تبدیل شوند می‌گویند که این قیود فعال یا اکتیو شده‌اند) پاسخ را می‌توان با حل معادله‌ی خطی زیر یافت:

$$\mathcal{L}\mathcal{v} = \nu$$

$$\mathcal{L} = \begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix}, \quad \mathcal{v} = \begin{bmatrix} z \\ \lambda \end{bmatrix}, \qquad \nu = \begin{bmatrix} -g \\ a \end{bmatrix} \tag{۲-۴۰}$$

که در عبارت‌های بالا $\lambda$ ضرایب لاگرانژ است و مقادیر متغیرهای $H$ و $A$ و $g$ و $a$ از تعریف مسئله‌ی بهینه سازی تعیین می‌گردند.

### ۲-۵-۱-۲- فرمول بندی با مجموعه‌ای کامل از متغیرها

اگر چه در اکثر مقالات فرمول بندی زیر در نظر گرفته نمی‌شود، ولی در حالت کلی فرض کنید بردار $z$ را به صورت زیر تعریف کنیم:

$$\begin{aligned} z^T = (u_k^T \quad \dots \quad & u_{k+N-1}^T \quad x_{k+1}^T \quad \dots \quad x_{k+N}^T \\ & y_k^T \quad \dots \quad y_{k+N-1}^T \quad e_k^T \quad \dots \quad e_{k+N-1}^T \\ & \Delta u_k^T \quad \dots \quad \Delta u_{k+N-1}^T) \end{aligned} \tag{۲-۴۱}$$

ماتریس $\mathcal{H}$ و بردار $g$ معادله‌ی

$$\min_z F(z) = \frac{1}{2} z^T \mathcal{H} z + g^T z + k$$

$$s.t. \mathcal{A}z = a$$

$$\mathcal{B}z \le b$$

$$z^l \le z \le z^u \tag{۲-۴۲}$$

را از این‌که باید $J_k$ در معادله‌ی (۲-۳۱) با $F(z)$ از معادله‌ی (۲-۴۲) برابر باشد بدست می‌آوریم. قیدهای $\mathcal{A}z = a$ مدل دینامیکی سیستم را که در معادله‌ی (۲-۲۹)و(۲-۳۰) است و تعاریف معادلات (۲-۳۷) و (۲-۳۸) را در بر دارد. برای مسئله‌ی کنترل پیش بینی که اینجا تعریف شده است، نامعادله‌ی $Bz \leq b$ صفر است درحالی‌که قیود فیزیکی و ایمنی را که پیش از این توضیح دادیم، در $z^u, z^l$ لحاظ کرده ایم.

در ادامه، تعاریفِ ماتریس‌هایی که در فرمول بالا به کار گرفته شده‌اند آورده می‌شود:

$$\mathcal{H} = diag(2(I_{N-1} \otimes R), 2(R+S), 0_{(N-1)\,dim_1 \bar{Q} \times (N-1).dim_2 \bar{Q}}, 0_{dim_1 Q \times N.dim\, y},$$

$$0_{N\, dim\, y \times N.dim\, y}, 2(I_N \otimes Q), 2(I_N \otimes S))$$

$$g = 0_{N.(2m+n+2r)x1}$$

$$\mathcal{A} = \begin{bmatrix} -(I_N \otimes B) & \mathcal{A}_{12} & 0 & 0 & 0 \\ -(I_N \otimes D) & \mathcal{A}_{22} & I_{N.r} & 0 & 0 \\ 0 & 0 & I_{N.r} & -I_{N.r} & 0 \\ \mathcal{A}_{41} & 0 & 0 & 0 & I_{N.m} \end{bmatrix} \qquad (۴۳\text{-}۲)$$

که در آن‌ها داریم:

**جدول۲-۱: مخفف ماتریس‌های به کار برده شده در معادلات**

| **Notation** |
| --- |
| $I_n \in \mathbb{R}^{n\times n}$ |
| $I_{n,k} = diag(1_{n-\|k\|\times 1}, k)$ |
| $0_{m\times n} \in \mathbb{R}^{m\times n}$ |
| $1_{m\times n} \in \mathbb{R}^{m\times n}$ |
| $A \in \mathbb{R}^{m\times n}: dim_1 A = m, dim_2 A = n$ |
| $A \otimes B$ |
| $diag(A_1, \dots, A_n)$ |
| $rot_{90} I_N$ |

ماتریس‌های $\mathcal{A}_{ij}$ مطابق زیر تعیین می‌شود:

**جدول۲-۲: ماتریس‌ها برای مجموعه متغیرها به صورت کامل**

| |
|---|
| $\mathcal{A}_{12} = I_{N.n} - I_{N,-1} \otimes A$ |
| $\mathcal{A}_{22} = -I_{N,-1} \otimes C$ |
| $\mathcal{A}_{41} = -I_{N.m} + I_{N,-1} \otimes I_m$ |

و داریم

$$a = (\begin{pmatrix} Ax_k \\ 0_{(N-1).n\times 1}\end{pmatrix}^T, \begin{pmatrix} Cx_k \\ 0_{(N-1).r\times 1}\end{pmatrix}^T, s^T, \begin{pmatrix} -u_{k-1} \\ 0_{(N-1)m\times 1}\end{pmatrix}^T)$$

$$z^l = \begin{pmatrix} 1_{N\times 1} \otimes u^l \\ -\infty.1^T_{N.n\times 1} \\ 1_{N\times 1} \otimes y^l \\ -\infty.1^T_{N.r\times 1} \\ 1_{N\times 1} \otimes \Delta u^l \end{pmatrix}, z^u = \begin{pmatrix} 1_{N\times 1} \otimes u^u \\ \infty.1^T_{N.n\times 1} \\ 1_{N\times 1} \otimes y^u \\ \infty.1^T_{N.r\times 1} \\ 1_{N\times 1} \otimes \Delta u^u \end{pmatrix} \quad (۲-۴۴)$$

بُعد متغیرهای مسئله‌ی برنامه ریزی درجه دوم حاصل به صورت $z = N.(n + 2m + 2r) \times 1$ و $a = N.(n + m + 2r) \times 1$ است. معمولاً اگر بردار $z$ را به صورت معادله‌ی (۲-۴۱) بیان کنیم، باعث می‌شود که ماتریس‌های $\mathcal{H},\mathcal{A}$ حاصل تُنُک باشند، برای اطلاعات بیشتر در مورد ماتریس‌های تُنُک به [۴۹] مراجعه شود.

### ۲-۵-۱-۳- تعیین فرمول‌ها با در نظر گرفتن مجموعه ای متوسط به عنوان متغیرها

از فرمول بندی کاملِ مسئله‌ی برنامه ریزی درجه دومی که در بخش پیش بدست آوردیم، $e_k, \Delta u_k, y_k$ را حذف می‌کنیم. بنابراین بردار مجهولات به صورت زیر خواهد بود:

$$z^T = (u_k^T \quad \dots \quad u_{k+N-1}^T \quad x_{k+1}^T \quad \dots \quad x_{k+N}^T) \quad (۲-۴۵)$$

با تعریف این بردار به عنوانِ بردار مجهولات، بردارها و ماتریس‌های مسئله‌ی برنامه ریزی درجه دوم حاصل به صورت زیر خواهند بود:

$$H = 2\begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22}\end{pmatrix}$$

$$g = \begin{pmatrix} 2(x_k^T C^T Q D - u_{k-1}^T S)^T \\ 0_{(N-1)m+Nn\times 1}\end{pmatrix}$$

$$\mathcal{A} = (-I_N \otimes B \quad I_{N.n} - I_{N,-1} \otimes A)$$

$$a = \begin{pmatrix} Ax_k \\ 0_{(N-1)n\times 1} \end{pmatrix}$$

$$z^l = \begin{pmatrix} 1_{N\times 1} \otimes u^l \\ -\infty.1_{N.n\times 1} \end{pmatrix}, \qquad z^u = \begin{pmatrix} 1_{N\times 1} \otimes u^u \\ \infty.1_{N.n\times 1} \end{pmatrix}$$

$$\mathcal{B} = \begin{pmatrix} \mathcal{B}_{11} & \mathcal{B}_{12} \\ -\mathcal{B}_{11} & -\mathcal{B}_{12} \\ \mathcal{B}_{13} & 0_{N.m\times N.n} \\ -\mathcal{B}_{13} & 0_{N.m\times N.n} \end{pmatrix}, \qquad b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} \tag{۴۶-۲}$$

که در ماتریس‌های بالا:

**جدول۲-۳:ماتریس‌ها برای مجموعه ای متوسط به عنوان متغیرها**

$$H_{11} = I_N \otimes (2S + R + D^T QD) - I_{N,-1} \otimes S - I_{N,+1} \otimes S$$

$$H_{12} = I_{N,+1} \otimes D^T QC = H_{21}^T$$

$$H_{22} = \begin{pmatrix} I_{N-1} \otimes C^T QC \\ \bar{Q} \end{pmatrix}$$

$$\mathcal{B}_{11} = I_N \otimes D, \;\; \mathcal{B}_{12} = I_{N,-1} \otimes C$$

$$\mathcal{B}_{31} = I_{N.m} - I_{N,-1} \otimes I_m$$

$$b_1 = \begin{pmatrix} y^u - Cx_k \\ 1_{N-1\times 1} \otimes y^u \end{pmatrix}, \;\; b_2 = \begin{pmatrix} -y^l + Cx_k \\ -1_{N-1\times 1} \otimes y^l \end{pmatrix}$$

$$b_3 = \begin{pmatrix} \Delta u^u + u_{k-1} \\ 1_{N-1\times 1} \otimes \Delta u^u \end{pmatrix}, \;\; b_4 = \begin{pmatrix} -\Delta u^l - u_{k-1} \\ -1_{N-1\times 1} \otimes \Delta u^l \end{pmatrix}$$

بُعد متغیرها و مجهولات اکنون به صورت $\dim z = N.(n+r) \times 1$ و $\dim a = N.n \times 1$ و همچنین $\dim b = N(r+m)$ است. تعریف بردار $z$ به صورت معادله‌ی (۲-۴۵) منجر به تُنُک شدن ماتریس‌های $H, \mathcal{A}, \mathcal{B}$ می‌شود.

### ۲-۵-۱-۴- -فرمول بندی مسئله با ساده‌ترین مجموعه‌ی متغیرها

رایج‌ترین شیوه‌ی فرمول بندی مسئله کنترل پیش بین به صورت برنامه ریزی درجه دوم به این صورت است که با استفاده از قیودِ مساوی، کلّیه‌ی مجهولات به استثنای ورودی‌های کنترلی آینده را حذف می‌کنیم، بنابراین:

$$z^T = (u_k^T \ldots u_{k+N-1}^T) \tag{۴۷-۲}$$

حال با این حالت دیگر قیود مساوی وجود نخواهند داشت بنابراین ماتریس‌ها و بردارها به صورت زیر هستند:

$$H = 2(\mathcal{H}_{N-1}(I_N \otimes Q)\mathcal{H}_{N-1} + (I_N \otimes R) + \Psi^T(I_N \otimes S)\Psi + \Omega + P^T C_k^T \bar{Q} C_k P)$$
$$g^T = 2(\mathcal{O}_N x_k - s)^T (I_N \otimes Q)\mathcal{H}_{N-1} + 2u_{k-1}^T L^T (I_N \otimes S)\Psi + 2x_k^T (A^N)^T \bar{Q} C_k P$$
$$\mathcal{B} = \begin{pmatrix} \Psi \\ -\Psi \\ \mathcal{H}_{N-1} \\ -\mathcal{H}_{N-1} \end{pmatrix}, \quad b = \begin{pmatrix} 1_{N\times1} \otimes \Delta u^u - Lu_{k-1} \\ -1_{N\times1} \otimes \Delta u^l + Lu_{k-1} \\ 1_{N\times1} \otimes y^u - \mathcal{O}_N x_k \\ -1_{N\times1} \otimes y^l + \mathcal{O}_N x_k \end{pmatrix}$$
$$z^l = 1_{N\times1} \otimes u^l, \quad z^u = 1_{N\times1} \otimes u^u \tag{۴۸-۲}$$

ماتریس‌هایی که در معادلات (۲-۴۸) مطرح شده‌اند را در جدول۲-۴ توضیح خواهیم داد. ابعاد ماتریس‌ها به صورت زیر است:

$$dim\, z = N.r \times 1, \quad dim\, b = 2N.(m + r) \times 1 \tag{۴۹-۲}$$

اما باید توجه داشت که با تعریف بردار $z$ به صورت معادله‌ی (۲-۴۷) ماتریس‌های $\mathcal{H}, \mathcal{B}$ چگال[1] می‌شوند.

**جدول۲-۴: ماتریس‌ها برای ساده‌ترین مجموعه متغیرها**

| |
|---|
| $\Psi = I_{N.m} - I_{N,-1} \otimes I_m$ |
| $L = \begin{pmatrix} -I_m \\ 0_{(N-1)\times m} \end{pmatrix}, \quad s^T = (s_k^T \cdots \; s_{k+N-1}^T)$ |
| $\Omega = \begin{pmatrix} 0_{N-1\times N-1} & 0_{N-1\times1} \\ 0_{1\times N-1} & 1 \end{pmatrix} \otimes S, P = rot_{90}(I_{N.m})$ |
| $\mathcal{C}_k = (B \;\; AB \;\; \ldots \; A^{k-1}B),$ |
| $\mathcal{O}_k^T = (\, C^T (CA)^T \;\; \ldots \;\; (CA^{k-1})T$ |
| $\mathcal{H}_k = \begin{pmatrix} D & 0_{dim\,D} & \cdots & 0_{dim\,D} \\ CB & D & \cdots & 0_{dim\,D} \\ \vdots & \vdots & \ddots & \vdots \\ CA^{k-1}B & CA^{k-2}B & \cdots & D \end{pmatrix}$ |

[1]Dense

### ۲-۵-۱-۵- فرمول بندی با استفاده از مجموعه‌ی متغیرهای ناشی از تجزیه کردن به عوامل $Q$ و $R$[1]

حذف قیود مساوی با استفاده از تجزیه کردن به عوامل $Q$ و $R$ می‌تواند انجام شود. اگر ماتریس‌های معرفی شده برای فرموله کردن کامل مسئله را با اندیس $c$ نشان دهیم داریم:

$$\mathcal{A}_c = \tilde{Q}\tilde{R} \tag{۲-۵۰}$$

که در رابطه‌ی بالا ماتریس $\tilde{Q}$ متعامد[2] است و ماتریس $\tilde{R}$ یک ماتریس بالا مثلثی است، و $\dim \tilde{R} = \dim \mathcal{A}_c$ بنابراین برای $\tilde{R}$ داریم:

$$\tilde{R} = (R_{11} \quad R_{12}) \tag{۲-۵۱}$$

که در این رابطه $\tilde{R}_{11}$ مربعی است و برای مسائل کنترل پیش بین قابل معکوس گیری است، همچنین $z_c$ به صورت زیر بخش بندی می‌شود:

$$z_c^T = (z_1^T \quad z_2^T) \tag{۲-۵۲}$$

که در آن $\dim z_1$ تعداد ستون‌ها در $R_{11}$ است و $\dim z_2 = N.m$. با این فرض داریم:

$$z_1 = R_{11}^{-1}(\tilde{Q}^T a_c - R_{12} z_2) \tag{۲-۵۳}$$

با حذف قیود مساوی، ماتریس‌های مسئله‌ی برنامه ریزی درجه دوم به صورت زیر تبدیل می‌شوند:

$$H = \begin{pmatrix} -R_{11}^{-1} \\ I \end{pmatrix}^T H_c \begin{pmatrix} -R_{11}^{-1} R_{12} \\ I \end{pmatrix}$$

$$g^T = \begin{pmatrix} R_{11}^{-1}\tilde{Q} a_c \\ 0 \end{pmatrix} H_c \begin{pmatrix} -R_{11}^{-1} R_{12} \\ I \end{pmatrix}$$

$$\mathcal{B} = \begin{pmatrix} \mathcal{B}_1 \\ -\mathcal{B}_1 \end{pmatrix}, \quad \mathcal{B}_1 = \begin{pmatrix} -R_{11}^{-1} R_{12} \\ I_{N.m} \end{pmatrix}$$

$$b = \begin{pmatrix} z_c^u - \begin{pmatrix} R_{11}^{-1}\tilde{Q} a_c \\ 0_{N.m\times N.m} \end{pmatrix} \\ -z_c^l + \begin{pmatrix} R_{11}^{-1}\tilde{Q}^T a_c \\ 0_{N.m\times N.m} \end{pmatrix} \end{pmatrix}$$

$$z_2^l = -\infty . 1_{N.m\times 1}, \quad z_2^u = \infty . 1_{N.m\times 1} \tag{۲-۵۴}$$

---

[1] QR Factorization

[2] Orthogonal

وقتی که $z_2$ را یافتیم، می‌توان با استفاده از معادله (۲-۵۳) مقدار $z_1$ را یافت. از آنجا که اولین مؤلفه‌ی $z_2$ برابرست با $\Delta u_k$ ، می‌توان $u_k$ را توسط $\Delta u_k + u_{k-1}$ یافت ، بنابراین در واقع نیازی به $z_1$ نداریم. ابعاد مسئله‌ی پایه‌ای کاهش یافته با تجزیه کردن به عوامل $Q$ و $R$ به صورت $\dim z_2 = N.m \times 1$ و $\dim b = 2.\dim z_c$ است. از آنجا که $\tilde{a}_c$ ممکن است با اندیس زمانی $k$ تغییر کند، ضروری است که $R_{11}^{-1}\tilde{Q}^T$ نیز بدست آورده شود که ابعاد آن $\dim a_c \times \dim a_c$ است. همچنین قابل ذکر است که این شیوه‌ی فرمول بندی منجر به ماتریس‌های $\mathcal{H}, \mathcal{B}$ چگال می‌شود.

# فصل ۳- بهینه سازی و روش‌های نوین آن

## ۳-۱- پیش‌گفتار

همان طور که در فصل اول و دوم گفته شد در کنترل پیش بین برای بدست آوردن سیگنال کنترلی نیاز به حل یک مسئله برنامه ریزی درجه دوم داریم. در این فصل، ابتدا مبانی ریاضی لازم شرح داده می‌شود. سپس از آنجا که الگوریتمی که در نهایت برای حل مسئله کنترل پیش بین توسعه داده شده است (الگوریتم اصلاح شده‌ی مهروترا) اصلاح شده ای از الگوریتم پیش‌گو-اصلاح‌گر مهروترا برای برنامه ریزی خطی است که توسط صلاحی و همکاران در [۴۰] صورت پذیرفته است، در ابتدا شرح کلی مسئله برنامه ریزی خطی و الگوریتم‌های پیش‌گو-اصلاح‌گر مهروترا و توسعه یافته‌ی آن توسط صلاحی و همکاران برای مسئله‌ی برنامه ریزی خطی می‌پردازیم، سپس به سراغ برنامه ریزی درجه دوم خواهیم رفت و در انتها نیز الگوریتم اصلاح شده‌ی مهروترا را برای برنامه ریزی درجه دوم گسترش خواهیم داد.

## ۳-۲- مقدمات ریاضی

### ۳-۲-۱- تحدب

تحدب[1] مفهومی پایه ای در بهینه سازی است. بسیاری از مسائل عملی این خاصیت را دارا هستند که آن‌ها را چه از لحاظ عملی و چه از لحاظ تئوری برای حل آسان‌تر می‌کند.

واژه‌ی "تحدب" را هم می‌توان به مجموعه‌ها و هم به توابع اختصاص داد. یک مجموعه‌ی $S \in \mathbb{R}^n$ یک مجموعه‌ی "محدب" است اگر پاره خطی که هر دو سر آن داخل $S$ است رسم کنیم کاملاً داخل مجموعه‌ی $S$ واقع شود.

به زبان ریاضی، برای هر دو نقطه $x, y \in S$ و $\alpha \in [0,1]$ داریم $\alpha x + (1-\alpha) y \in S$.

تابع $f$ را روی مجموعه‌ی محدب $S$، یک تابع محدب گویند هرگاه برای هر دو نقطه‌ی $x, y \in S$ و $\alpha \in [0,1]$ داشته باشیم:

$$f(\alpha x + (1-\alpha) y) \le \alpha f(x) + (1-\alpha) f(y) \qquad (۱-۳)$$

[1] Convexity

مثال‌های ساده‌ی مجموعه‌های محدب شامل دایره‌ی واحد و چند وجهی‌ها هستند، لازم به ذکر است که یک چند وجهی را می‌توان با معادله‌ها و نامعادله های خطی به صورت زیر توصیف کرد:

$$\{x \in \mathbb{R}^n | Ax = b, Cx \ \leq d\} \tag{۳-۲}$$

که در آن $C$ و $A$ ماتریس‌هایی با ابعاد مناسب و $b$ و $d$ بردارهای معلوم هستند. مثال‌هایی ساده از توابع محدب تابع آفینی زیر است:

$$f(x) = c^T x + \alpha \tag{۳-۳}$$

که در آن $c \in \mathbb{R}^n$ و اسکالر $\alpha$ داده شده اند.

همچنین تابع درجه دوم همگن $f(x) = x^T Hx$ محدب است اگر و تنها اگر ماتریس $H$ نیمه مثبت معین (متقارن) باشد.

## ۳-۲-۲- بهینه سازی نامقید

### ۳-۲-۲-۱- بهینه سازی موضعی و سراسری[۱]

به طور معمول اکثر الگوریتم‌های بهینه سازی، نقطه‌ی مینیمم موضعی را نتیجه می‌دهند به این معنی که در آن نقطه تابع هزینه از تمامی نقاط شدنی در همسایگی آن مقدار کمتری را دارد. این الگوریتم‌ها نقطه‌ی مینیمم سراسری را که نقطه ای با کمترین مقدار تابع هزینه در "کل فضای شدنی" است، در مسائل خاص پیدا می‌کنند. به طور دقیق‌تر، در صورتی‌که مسئله محدب باشد می‌توان مطمئن بود که نقطه‌ی مینیمم کننده‌ی موضعی پیدا شده، نقطه‌ی مینیمم کننده‌ی سراسری است. بیان ریاضی مفاهیم مینیمم‌کننده‌ی موضعی و سراسری به صورت زیر هستند:

- نقطه‌ی $x^*$ مینیمم موضعی است، هر گاه برای $x$ در دامنه‌ی $f$، داشته باشیم: $f(x^*) \leq f(x)$

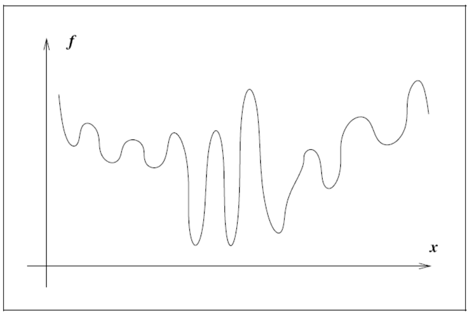
- نقطه‌ی $x^*$ مینیمم موضعی ضعیف[۲] است هرگاه یک همسایگی $\mathcal{N}$ حول $x^*$ وجود داشته باشد به طوری که برای همه‌ی $x \in \mathcal{N}$ داشته باشیم: $f(x^*) \leq f(x)$.

- نقطه‌ی $x^*$ نقطه‌ی مینیمم موضعی اکید (مینیمم موضعی قوی) است هرگاه یک همسایگی از $\mathcal{N}$ حول $x^*$ وجود داشته باشد به طوری که برای همه‌ی $x^* \neq x \in \ \mathcal{N}$ داشته باشیم: $f(x^*) < f(x)$

[1]Local & Global Optimization
[2]Weak Local Minimizer

- نقطه‌ی $x^*$ نقطه‌ی مینیمم موضعی تنها[1] است هرگاه یک همسایگی $\mathcal{N}$ حول $x^*$ وجود داشته باشد به طوری که $x^*$ تنها مینیمم موضعی در آن همسایگی باشد.

شکل۳-۱ تابعی با چندین نقطه‌ی مینیمم موضعی را نشان می‌دهد، برای چنین توابعی معمولاً یافتن نقاط مینیمم سراسری سخت است، زیرا الگوریتم‌ها متمایل به برقرار شرایط لازم بهینگی هستند که این امر در مینیمم کننده‌های موضعی نیز اتفاق میافتد. در مسائل بهینه سازی که مربوط به ترکیب مولکولی هستند تابع هدف برای بهینه سازی ممکن است نقاط مینیمم موضعی متعددی داشته باشد[۲۳].




**شکل۳-۱: تابعی با نقاط مینیمم متعدد (عکس از منبع [۲۳] صفحه‌ی ۱۳)**

### ۳-۲-۲- شرایط لازم و کافی بهینگی در مسائل نامقید [۲۳]

در این بخش شرایط لازم بهینگی را برای توابع هموار معرفی می‌کنیم. فرض کنید $f$ تابعی دوبار به طور پیوسته مشتق پذیر است. حال، می‌توان درباره‌ی وضعیت نقطه‌ی مینیمم کننده با استفاده از گرادیان[2] تابع، یعنی $\nabla f(x^*)$ و هسین[3] آن، یعنی $\nabla^2 f(x^*)$ اظهار نظر کرد.

#### شرایط لازم مرتبه اول

فرض کنید $x^*$ مینیمم موضعی تابع دوبار مشتق پذیر $f$ است. در این صورت: $\nabla f(x^*) = 0$.

[1]Isolated Local Minimizer
[2]Gradient
[3]Hessian

**شرایط لازم مرتبه دوم**

اگر $x^*$ یک نقطه‌ی مینیمم موضعی برای $f$ باشد و $\nabla^2 f$ در یک همسایگی باز حول $x^*$ پیوسته باشد، آنگاه $\nabla f(x^*) = 0$ و $\nabla^2 f(x^*)$ نیمه معین مثبت است.

اکنون شرایط کافی بهینگی را بر تابع دوبار به طور پیوسته مشتق‌پذیر $f$ در نقطه‌ی $x^*$ بیان می‌کنیم.

**شرایط کافی مرتبه دوم**

فرض کنید $\nabla^2 f$ در یک همسایگی باز از $x^*$ پیوسته بوده و $\nabla f(x^*) = 0$ و $\nabla^2 f(x^*)$ معین مثبت باشد. در این صورت، $x^*$ نقطه‌ی مینیمم موضعی اکید تابع $f$ است.
شرط بالا شرط لازم نیست بلکه شرط کافی است. به عنوان مثال، برای تابع $f(x) = x^4$، نقطه‌ی $x^* = 0$ یک نقطه‌ی مینیمم موضعی اکید است ولی هِسین آن صفر است، به عبارت دیگر، مثبت معین نیست.

**قضیه ۳-۱:**

برای تابع محدب $f$، هر مینیمم موضعی $x^*$، مینیمم سراسری برای تابع استبه علاوه، شرایط لازم مرتبه‌ی اول بهینگی، شرایط کافی نیز هستند.

برای اثبات قضیه‌ی فوق به [۲۳] مراجعه شود.

## ۳-۲-۳- شرایط لازم و کافی بهینگی مسائل مقید[۲۳]

مسئله‌ی بهینه سازی مقید زیر را در نظر بگیرید:

$$\min f(x) \;\; subject\ to \begin{cases} c_i(x) = 0, & i \in \mathcal{E} \\ c_i(x) \geq 0, & i \in \mathcal{I} \end{cases} \tag{۳-۴}$$

که در آن $\mathcal{E}$ مجموعه اندیس قیود تساوی، $\mathcal{I}$ مجموعه انیس قیود نامساوی، $c_i: \mathbb{R}^n \to \mathbb{R}$، توابع قیود و $f: \mathbb{R}^n \to \mathbb{R}$ تابع هدف است. شرایط بهینگی به صورت زیر برای تابع مقید تعریف می‌شود:

فرض کنید $\Omega$ مجموعه جواب‌های شدنی مسئله‌ی (۳-۴) است، یعنی:

$\Omega = \{x \in \mathbb{R}^n | c_i(x) = 0\,, i \in \mathcal{E}, c_i(x) \geq 0, i \in J\}$ ، در این صورت داریم:

- تابع $x^*$ را مینیمم موضعی برای مسئله‌ی $\min_{x\in\Omega} f(x)$ می‌نامند هرگاه $x^* \in \Omega$ و یک همسایگی $\mathcal{N}$ از $x^*$ وجود داشته باشد به طوری که $f(x) \geq f(x^*)$ برای هر $x \in \mathcal{N} \cap \Omega$

- بردار $x^*$ را مینیمم کننده‌ی موضعی اکید برای مسئله (۳-۴) می‌نامند، هرگاه $x^* \in \Omega$ و یک همسایگی $\mathcal{N}$ از $x^*$ وجود داشته باشد به طوری که $f(x) > f(x^*)$ ، برای همه‌ی $x \in \mathcal{N} \cap \Omega$ و $x \neq x^*$.

**شرایط لازم بهینگی مرتبه اول**

پیش از شروع بحث تابع لاگرانژی را به صورت زیر تعریف می‌کنیم:

$$\mathcal{L}(x,\lambda) = f(x) - \sum_{i\in\mathcal{E}\cup\mathcal{I}} \lambda_i c_i(x) \tag{۳-۵}$$

در این صورت، شرایط لازم بهینگی مرتبه اول به صورت زیر بیان می‌شود:

فرض کنید که $x^*$ مینیمم کننده‌ی موضعی مسئله‌ی

$$\min f(x)\ \ subject\ to \begin{cases} c_i(x) = 0, & i \in \mathcal{E} \\ c_i(x) \geq 0, & i \in \mathcal{I} \end{cases} \tag{۳-۶}$$

است. همچنین فرض کنید توابع $f$ و $c_i$ها همگی به طور پیوسته مشتق‌پذیر هستند. اگر شرایط قیدی $LICQ$ [۲۳] در نقطه $x^*$ برقرار باشد، آنگاه بردار ضریب لاگرانژی $\lambda^*$ با مؤلفه های $\lambda_i^*$، برای $i \in \mathcal{E} \cup \mathcal{I}$، وجود دارد به طوری که شرایط زیر در نقطه‌ی $(x^*,\lambda^*)$ ارضا می‌شود:

$$\begin{aligned} \nabla_x \mathcal{L}(x^*,\lambda^*) &= 0, \\ c_i(x^*) &= 0, \qquad for\ all\ i \in \mathcal{E} \\ c_i(x^*) &\geq 0, \qquad for\ all\ i \in \mathcal{I}, \\ \lambda_i^* &\geq 0, \qquad for\ all\ i \in \mathcal{I}, \\ \lambda_i^* c_i(x^*) &= 0, \qquad for\ all\ i \in \mathcal{E} \cup \mathcal{I} \end{aligned} \tag{۳-۷}$$

این شرایط به کاروش-کیون-تاکر (KKT) معروف هستند. شرط $\lambda_i^* c_i(x^*) = 0$ را شرایط مکملی[۱] می‌نامند و مفهوم این شرط این است که یا قید $i$ ام فعال است، یا $\lambda_i^* = 0$ یا هر دو در نقطه‌ی $x^*$ برقرار هستند.

**شرایط لازم مرتبه دوم**: فرض کنید $x^*$ مینیمم کننده‌ی موضعی مسئله (۳-۴) باشد و شرایط LICQ در $x^*$ برقرار است. هم‌چنین، فرض کنید شرایط KKT در $(x^*,\lambda^*)$ برقرار هستند. در این صورت، داریم:

---

[1]Complementary

$$\mathcal{w}^T \nabla^2_{xx}\mathcal{L}(x^*,\lambda^*)\mathcal{w} \geq 0, \quad \forall\, \mathcal{w} \in \mathcal{C}(x^*,\lambda^*) \tag{۸-۳}$$

که در آن

$$\mathcal{C}(x^*,\lambda^*) = \{\mathcal{w} \in \mathcal{F}(x^*) | \nabla c_i(x^*)^T\mathcal{w} = 0, \forall\, i \in \mathcal{A}(x^*) \cap \mathcal{I}\; with\; \lambda_i^* > 0\} \tag{۹-۳}$$

و $\mathcal{A}(x^*)$ مجموعه قیود فعال در $x^*$ هستند. مجموعه‌ی $\mathcal{C}(x^*,\lambda^*)$ را مخروط بحرانی[۱] در $x^*$ گویند.

**تعریف (مجموعه‌ی فعال)[۲]**

مجموعه‌ی فعال$\mathcal{A}(x)$ در هر نقطه‌ی شدنی[۳] $x$، از مجموع اندیس‌های قیدهای تساوی به همراه قیدهای نامساوی $i \in \mathcal{I}$ که در آن‌ها $c_i(x) = 0$ است، تشکیل شده است،یعنی

$$\mathcal{A}(x) = \mathcal{E} \cup \{i \in \mathcal{I} | c_i(x) = 0\} \tag{۱۰-۳}$$

**شرایط قیدی LICQ**

در نقطه‌ی شدنی $x$ گوییم شرایط قیدهای خطی مستقل[۴] یا LICQ برقرار است هرگاه مجموعه‌ی گرادیان های قیود فعال، یعنی $\{\nabla c_i(x), i \in \mathcal{A}(x)\}$ ، مستقل خطی باشند.

**شرایط کافی مرتبه دوم**

فرض کنید در نقطه‌ی شدنی $x^* \in \mathbb{R}^n$ برای مسئله‌ی (۳-۴)، بردار ضرایب لاگرانژ$\lambda^*$ موجود باشد که در شرایط KKT صدق کند و داشته باشیم :

$$\mathcal{w}^T \nabla^2_{xx}\mathcal{L}(x^*,\lambda^*)\mathcal{w} > 0, \quad \forall \mathcal{w} \in \mathcal{C}(x^*,\lambda^*), \mathcal{w} \neq 0 \tag{۱۱-۳}$$

در این صورت $x^*$ یک مینیمم کننده‌ی موضعی اکید برای مسئله‌ی (۳-۴) است.

[1]Critical Cone
[2]Active Set
[3]Feasible
[4]Linear Independence Constraint Qualification(LICQ)

**شرایط لازم و کافی بهینگی برای مسائل برنامه ریزی درجه دوم مقید**

برای مسئله‌ی برنامه ریزی درجه دوم، تابع لاگرانژی عبارت است از

$$\mathcal{L}(x,\lambda)=\frac{1}{2}x^TGx+x^Tc-\sum_{i\in\mathcal{I}\cup\mathcal{E}}\lambda_i(a_i^Tx-b_i) \tag{۱۲-۳}$$

همچنین مجموعه‌ی فعال $\mathcal{A}(x^*)$ که تشکیل شده است از اندیس قیودی که در نقطه‌ی $x^*$ برای آن‌ها مساوی صدق می‌کند (و نه نامساوی).

$$\mathcal{A}(x^*)\{i\in\mathcal{I}\cup\mathcal{E}|a_i^Tx^*=b^i\} \tag{۱۳-۳}$$

شرایط KKT برای این مسئله منجر به معادلات زیر می‌شود:

$$\begin{aligned}
&Gx^*+c-\sum_{i\in\mathcal{A}(x^*)}\lambda_i^*a_i=0,\\
&a_i^Tx^*=b_i,\qquad \forall\, i\in\mathcal{A}(x^*)\\
&a_i^Tx^*\geq b_i,\qquad \forall\, i\in\mathcal{I}\backslash\mathcal{A}(x^*)\\
&\lambda_i^*\geq 0,\qquad \forall\, i\in\ \mathcal{I}\cap\mathcal{A}(x^*)
\end{aligned} \tag{۱۴-۳}$$

برای مسائل برنامه ریزی درجه دوم محدب، یعنی وقتی که $G$ نیمه معین مثبت است، شرایط لازم KKT فوق، شرایط کافی نیز هستند که ایجاب می‌کند $x^*$ مینیمم سراسری برای تابع باشد.

در دیدگاهی دیگر، شرایط کافی برای آنکه $x^*$ مینیمم موضعی باشد این است که ماتریس $Z^TGZ$ معین مثبت باشد، که در آن ماتریس $Z$ ماتریس پایه‌ای برای پوچی ماتریس ژاکوبین قیود فعال است یعنی ماتریسی که سطرهایش برای همه‌ی $i\in\mathcal{A}(x^*)$ مقدار $a_i^T$ است.

لازم به ذکر است که اگر $G$ معین مثبت نباشد، آنگاه مسئله‌ی برنامه ریزی درجه دوم ممکن است بیش از یک نقطه‌ی مینیمم موضعی اکید داشته باشد.

## ۳-۳- روش‌های نقطه‌ی درونی اولیه-دوگان برای برنامه ریزی خطی[۳۲]

در این بخش، ابتدا مفاهیم روش‌های نقطه‌ی درونی اولیه-دوگان را برای مسئله‌ی برنامه ریزی خطی بیان می‌کنیم. سپس الگوریتم پیش‌پو-اصلاح‌گر مهروترا و اصلاح شده‌ی آن توسط صلاحی و همکاران در [۴۰] را مورد بررسی قرار می‌گیرد. تعمیمی از این روش‌ها به برنامه ریزی درجه دوم در بخش‌های آتی ارائه خواهد شد.

فرم استاندارد یک مسئله‌ی برنامه ریزی خطی به صورت زیر تعریف می‌شود:

$$\min c^T x\,, subject\ to\ Ax = b, x \geq 0 \qquad (۱۵-۳)$$

که در آن $c$ و $x$ بردارهایی در فضای $\mathbb{R}^n$، $b$ برداری در فضای $\mathbb{R}^m$ و $A$ ماتریسی $m \times n$ با رتبه‌ی سطری کامل است. دوگان مسئله‌ی (۳-۱۵)به صورت زیر است:

$$\max b^T \lambda\,, subject\ to\ \ A^T\lambda + s = c, s \geq 0, \qquad (۱۶-۳)$$

که در آن $\lambda \in \mathbb{R}^m$ و $s \in \mathbb{R}^n$ .شرایط لازم (و در نتیجه کافی) بهینگی برای مسائل اوله و دوگان فوق(شرایطKKT) به صورت زیر هستند:

$$\begin{gathered} A^T\lambda + s = c \\ Ax = b \\ (x,s)x_i s_i = 0, \qquad i = 1,2,\dots,n\ 0 \\ (x,s) \geq 0 \end{gathered} \qquad (۱۷-۳)$$

در روش اولیه-دوگان، معمولاً یک معیار برای مطلوبیت نقطه‌ها در فضای جستجو، معیار دوگانی[1] است که به صورت زیر تعریف می‌شود:

$$\mu = \frac{1}{n}\sum_{\mathrm{i}=1}^{\mathrm{n}} x_i s_i = \frac{x^T s}{n} \qquad (۱۸-۳)$$

روش‌های اولیه-دوگان جواب دستگاه (۳-۱۷) را با اِعمال روش نیوتون پیدا می‌کنند. در نقطه‌ی جاری $(x,\lambda,s)$، جهت جستجوی نیوتن برای دستگاه (۳-۱۷) از حل دستگاه زیر حاصل می‌شود:

$$J(x,\lambda,s)\begin{bmatrix}\Delta x\\ \Delta\lambda\\ \Delta s\end{bmatrix} = -F(x,\lambda,s), \qquad (۱۹-۳)$$

[1] Duality Measure

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe \end{bmatrix} \qquad (۲۰-۳)$$

که در آن عموماً یک گام کامل در راستای این جهت باعث می‌شود که قید $(x, s) \geq 0$ نقض شود. بنابراین، در راستای جهت نیوتون طول گام $\alpha$ طوری تعیین می‌شود که این قید نقض نشود، یعنی $\alpha \in (0,1]$ طوری تعیین می‌شود که

$$(x, \lambda, s) + \alpha(\Delta x, \Delta \lambda, \Delta s) \geq 0 \qquad (۲۱-۳)$$

بنابراین حرکت در جهت نیوتون که گاهی به آن جهت مقیاس بندی آفینی[۱] هم می‌گویند، عموماً چندان باعث پیشرفت سریع در یافتن جواب مسئله نمی‌شود.

## مسیر مرکزی[۲]

مجموعه‌ی شدنی[۳] $\mathcal{F}$ و شدنی اکید[۴] $\mathcal{F}^0$ را به صورت زیر تعریف می‌کنیم:

$$\mathcal{F} = \{(x, \lambda, s) | Ax = b, A^T\lambda + s = c, (x, s) \geq 0\} \qquad (۲۲-۳)$$

$$\mathcal{F}^0 = \{(x, \lambda, s) | Ax = b, A^T\lambda + s = c, (x, s) > 0\} \qquad (۲۳-۳)$$

مسیر مرکزی $\mathcal{C}$ منحنی است از نقاط اکیداً شدنی که آن‌را با استفاده از اسکالر $\tau > 0$ نشان می‌دهند، و هر نقطه از آن $(x_\tau, \lambda_\tau, s_\tau) \in \mathcal{C}$ در معادلات زیر صدق می‌کنند:

$$\begin{aligned} A^T\lambda + s &= c, \\ Ax &= b, \\ x_i s_i &= \tau, \qquad i = 1,2, \dots, n \\ (x, s) &> 0 \end{aligned} \qquad (۲۴-۳)$$

---

[1]Affine scaling direction
[2]Central Path
[3]Feasible Set
[4]Strictly feasible set

در اینجا همان‌طور که قابل مشاهده است در معادله‌ی سوم در سمتِ راست معادله به جای $0$ متغیر $\tau$ قرار گرفته است. پس مسیر مرکزی به صورت زیر تعریف می‌شود:

$$\mathcal{C} = \{(x_\tau, \lambda_\tau, s_\tau) | \tau > 0\} \qquad (۲۵\text{-}۳)$$

اگر $\mathcal{F}^0$ تهی نباشد برای هر $\tau > 0$ یک نقطه‌ی یکتا $(x_\tau, \lambda_\tau, s_\tau)$ تعریف می‌شود. هر چه $\tau$ به سمت صفر میل کند، طبیعتاً معادلاتی که در (۳-۲۴) نوشته شد به (۳-۱۷) نزدیک‌تر می‌شود، وقتی که $\tau$ به سمت صفر میل می‌کند، مسئله به سمت پاسخ مسئله‌ی برنامه ریزی خطی همگرا می‌شود. بنابراین مسیر مرکزی با مثبت نگه داشتن دائمی $x_i s_i$ از ورود مسئله به مسیرهای انحرافی جلوگیری می‌کند و با میل کردن این حاصل ضرب به صورت تدریجی به صفر به مسئله‌ی اصلی نزدیک می‌شود.

روش‌های مسیریاب اولیه-دوگان[1] تکرارهای الگوریتم خود را در همسایگی‌ای از مسیر مرکزی نگه می‌دارند و $\mathcal{C}$ را تا رسیدن به پاسخ مسئله دنبال می‌کنند. به تدریج با کمتر کردن $\mu_k$ هر چه که $k$ بزرگ‌تر می‌شود، مطمئن می‌شویم که تکرارهای الگوریتم بیشتر به شرایط KKT و معادلات نوشته شده در (۳-۱۷) نزدیک می‌شود.

دو مجموعه بیشتر از بقیه به عنوان همسایگی‌های $\mathcal{C}$ تعریف می‌شوند:

$$\mathcal{N}_2(\theta) = \{(x, \lambda, s) \in \mathcal{F}^0 | \|XSe\ \mu e\|_2 \leq \theta\mu\} \qquad (۲۶\text{-}۳)$$

برای $\theta \in [0,1)$ و همچنین

$$\mathcal{N}_{-\infty}(\gamma) = \{(x, \lambda, s) \in \mathcal{F}^0 | x_i s_i \geq \gamma\mu \qquad all\ i = 1,2,3,\ldots,n\} \qquad (۲۷\text{-}۳)$$

برای $\gamma \in (0,1]$

### ۳-۳-۱- روش پیش گو-اصلاح گر[2]

روش پیش گو-اصلاح گر، دو نوع گام مختلف برای رسیدن به مقصد طی می‌کند، تکرارهای این روش همگی این الگوریتم را در فضای بین دو گام پیش گو و اصلاح گر نگاه می‌دارد:

**گام پیش گو**: که در آن برای آنکه $\mu$ کاهش پیدا کند $\sigma_k = 0$

**گام اصلاح گر:** که در آن برای بهبود و نزدیک‌تر بودن به مرکزیت[3] باید $\sigma_k = 1$

---

[1]Primal-Dual Path following algorithms

[2]Predictor-Corrector Algorithm

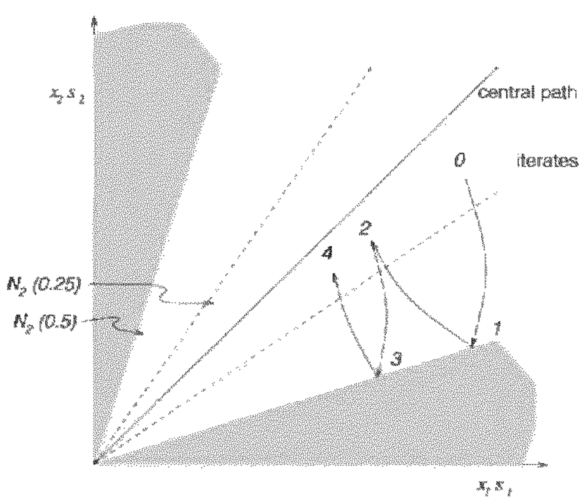
[3]Centrality

دلیل نام گذاری این روش به خاطر شباهت این روش با روشی به همین نام در معادلات دیفرانسیل معمولی (ODE) است.

این روش دارای دو همسایگی است که یکی در داخل دیگری واقع است، در این روش تکرار شونده تکرارهای با اندیس زوج ($(x^k, \lambda^k, s^k)$ که $k$ در آن زوج است) در داخل همسایگی درونی و تکرارهای با اندیس فرد که در آن $k$ فرد است در داخل همسایگی بیرونی قرار می‌گیرند. برای مثال دو تکرار اولیه از این روش به صورت زیر است:

نقطه‌ی شروع را $(x^0, \lambda^0, s^0)$ می‌نامیم که در داخل همسایگی درونی است، با قرار دادن $\sigma_0 = 0$ گام پیش گو را محاسبه می‌کنیم، در جهت مشخص شده تا جایی که به مرز همسایگی بیرونی برسیم حرکت می‌کنیم، در این نقطه توقف می‌کنیم و آنگاه گام اصلاح گر $(x^1, \lambda^1, s^1)$ را تعریف می‌کنیم. برای محاسبه‌ی گام اصلاح گر، $\sigma_1 = 1$ قرار می‌دهیم. یک گام کامل در این جهت $(\alpha = 1)$ منجر به تکرار جدید $(x^2, \lambda^2, s^2)$ خواهد شد که مجدداً به داخل همسایگی درونی باز خواهد گشت. این چرخه که شامل دو گام می‌شود را به صورت تکرار شونده تکرار می‌کنیم که در هر گام، گام پیش گو (با اندیس‌های تکرار فرد) بر روی مرز همسایگی بیرونی قرار می‌گیرند و گام‌های اصلاح گر (که دارای اندیس زوج هستند) در داخل همسایگی درونی قرار می‌گیرند.

گام پیش گو مقدار $\mu$ را با فاکتور $(1 - \alpha)$ کاهش می‌دهد که $\alpha$ طول گام است. گام‌های اصلاح گر مقدار $\mu$ را بدون تغییر رها می‌کنند حال آنکه با بازگشتن به داخل همسایگی درونی به الگوریتم فضای بیشتری می‌دهند که در گام پیش بین بعدی پیشروی کند. برای مثال اگر همسایگی درونی را $\mathcal{N}_2(0.25)$ و همسایگی بیرونی را $\mathcal{N}_2(0.5)$ در نظر بگیریم، الگوریتم پیش گو-اصلاح گر را می‌توان به صورت جدول۳-۱و شکل۳-۲ خلاصه نمود:




**شکل۳-۲: تکرارهای روش‌های اولیه-دوگان در مختصات xs برگرفته از[۳۲]**

**جدول ۳-۱: خلاصه‌ی الگوریتم پیش گو اصلاح گر**

**الگوریتم پیش گو اصلاح گر:**

فرض کنیم نقطه‌ی شروع $(x^0, \lambda^0, s^0) \in \mathcal{N}_2(0.25)$ داده شده است

**برای** $k = 0,1,2,...$

**اگر** $k$ زوج است

**❋گام پیش گو❋**

با فرض $\sigma_k = 0$ معادله‌ی

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^T & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta\lambda^k \\ \Delta s^k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -X^k S^k e + \sigma_k \mu_k e \end{bmatrix}, where\ \mu_k = (x_k)^T s^k / n$$

را برای بدست آوردن $(\Delta x^k, \Delta\lambda^k, \Delta s^k)$ حل کنید

$\alpha_k$ را به عنوان بزرگ‌ترین $\alpha$ در فاصله‌ی $[0,1]$ طوری بدست آورید که:

$$\left(x^k(\alpha), \lambda^k(\alpha), s^k(\alpha)\right) \in \mathcal{N}_2(0.5)$$

قرار دهید: $(x^{k+1}, \lambda^{k+1}, s^{k+1}) = \left(x^k(\alpha_k), \lambda^k(\alpha_k), s^k(\alpha_k)\right);$

**در غیر این صورت**

**❋گام اصلاح گر❋**

با فرض $\sigma_k = 1$ معادله‌ی

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^T & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta\lambda^k \\ \Delta s^k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -X^k S^k e + \sigma_k \mu_k e \end{bmatrix}, where\ \mu_k = (x_k)^T s^k / n$$

را برای بدست آوردن $(\Delta x^k, \Delta\lambda^k, \Delta s^k)$ حل کنید

قرار دهید: $(x^{k+1}, \lambda^{k+1}, s^{k+1}) = (x^k, \lambda^k, s^k) + (\Delta x^k, \Delta\lambda^k, \Delta s^k);$

**پایان(اگر)**

**پایان(برای)**

### ۳-۲-۳- الگوریتم مهروترا[۲۳]

این الگوریتم که به الگوریتم نقطه میانیِ اولیه-دوگان پیش گو-اصلاح گر مهروترا معروف است عملی‌ترین الگوریتمی است که در زمینه‌ی الگوریتم‌های اولیه-دوگان ارائه شده است. قوّت این روش در این است که به خوبی پیاده سازی شده است و اکثرِ بسته های نرم افزاری که نوشته شده است با استفاده از این الگوریتم توسعه یافته است.

الگوریتمِ پیش‌گو-اصلاح‌گر مهروترا با الگوریتمِ کلیِ روش‌های اولیه-دوگان کمی متفاوت است. در این الگوریتم، روش نیوتون مانندی برای یافتن جهت جستجو انجام می‌پذیرد. ضمناً در این روش پارامتر مرکزیت[1] $(\sigma)$ در هر گام تغییر می‌کند. در روش مهروترا یک دنباله از تکرارها بوجود می‌آید که در آن $(x^k, s^k) > 0$ است.

جهت جستجوی هر تکرار از سه مؤلفه‌ی زیر تشکیل شده است:

۱- یک جهت مقیاس بندیِ آفینی پیش گو[2] که توسط جهتِ نیوتون و با فرمول زیر داده می‌شود:

$$F(x,\lambda,s) = \begin{bmatrix} A^T\lambda + s - c \\ Ax - b \\ XSe \end{bmatrix} = 0$$

$$(x,s) \geq 0$$

$$where\ X = diag(x_1, x_2, \dots, x_n), \qquad S = diag(s_1, s_2, \dots, s_n)$$

$$e = (1,1,\dots,1)^T \qquad (۳-۲۸)$$

۲- یک جمله که به آن جمله‌ی مرکزی می‌گویند و اندازه‌ی آن توسط پارامتر مرکزیت $\sigma$ تعیین می‌شود.

۳- یک جهت " اصلاح گر" که سعی در جبران سازی بعضی خواص غیرخطی در جهت مقیاس بندی آفینی می‌کند.

در الگوریتمِ مهروترا مؤلفه‌ی مقیاس بندیِ آفینی پیش از مؤلفه‌ی مرکزی و مستقل از آن بدست می‌آید. اگر جهت مقیاس بندیِ آفینی در کاهش معیار دوگانی $\mu$ باعث پیشرفت مناسبی شود، در عین این که پارامترهای قید را خدشه دار نمی‌کند (یعنی $(x,s) > 0$) به این نتیجه می‌توان رسید که در این گام نیاز کمی به پارامتر"مرکزیت" حس می‌شود، بنابراین می‌توان مقدار کوچکی را برای σ در نظر گرفت.

[1]Centring Parameter

[2]Affine-Scaling Predictor Direction

اما، اگر با کمی حرکت در جهت مقیاس بندی آفینی، به مرزِ$(x,s) > 0$ رسیدیم، به این نتیجه می‌رسیم که احتیاج به "مرکزیت" بخشیدن به الگوریتم خیلی زیاد حس می‌شود، بنابراین σ را برابر یک قرار می‌دهیم $\sigma = 1$ .

**الگوریتم مهروترا**

فرض کنید که نقطه‌ی جاری $(x,\lambda,s)$ داده شده است، جهت مقیاس بندی آفینی با حلِ سیستم زیر بدست می‌آید:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix}\begin{bmatrix} \Delta x^{aff} \\ \Delta \lambda^{aff} \\ \Delta s^{aff} \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe \end{bmatrix}$$

$$r_b = Ax - b, \qquad r_c = A^T\lambda + s - c \tag{۳-۲۹}$$

همان‌طور که مشخص است در بدست آوردن این سیستم مقدار $\sigma = 0$ قرار داده شده است. اکنون در این جهت باید طول گام‌ها را تا مرز مشخص کنیم، برای این منظور باید محاسبه های مجزا برای مسئله‌ی اولیه و مسئله‌ی دوگان انجام بدهیم.

$$\alpha_{aff}^{pri} = \arg\max\{\alpha \in [0,1] \big| x + \alpha\Delta x^{aff} \geq 0\} \tag{۳-۳۰}$$

$$\alpha_{aff}^{dual} = \arg\max\{\alpha \in [0,1] \big| s + \alpha\Delta s^{aff} \geq 0\} \tag{۳-۳۱}$$

برای اینکه معیاری برای ارزیابی جهت مقیاس بندیِ آفینی داشته باشیم، $\mu_{aff}$ را تعریف می‌کنیم:

$$\mu_{aff} = \left(x + \alpha_{aff}^{pri}\Delta x^{aff}\right)^T (s + \alpha_{aff}^{dual}\Delta s^{aff})/n \tag{۳-۳۲}$$

اگر $\mu_{aff} \ll \mu$ آنگاه جهت مقیاس بندیِ آفینی مناسب است و منجر به کاهش شدید $\mu$ خواهد شد. در این حالت همان‌طور که گفته شد مقدار $\sigma$ را نزدیک به صفر انتخاب می‌کنیم. اگر $\mu_{aff}$ کمی کوچک‌تر از $\mu$ باشد، مقدار $\sigma$ را نزدیک به ۱ انتخاب می‌کنیم. این کار باعث می‌شود که به مسیر مرکزی نزدیک شویم و بنابراین در تکرار بعدی به خوبی $\mu$ را کاهش می‌دهد. مهروترا در [۳۳] ابتکار[1] زیر را که بسیار کارآمد است پیشنهاد کرده است.

$$\sigma = \left(\frac{\mu_{aff}}{\mu}\right)^3 \tag{۳-۳۳}$$

[1]Heuristic

برای اینکه مؤلفه‌ی گام مرکزی را محاسبه کنیم، سیستم خطی را در حالتی که سمت راست معادله‌ی آن مقدار $(0,0,\sigma\mu e)$ قرار دارد محاسبه می‌کنیم.

وقتی که یک گام طی شد، ضرب مؤلفه ای (هادامارد)[1] $x_i s_i$ به عبارت $\Delta x_i^{aff}\Delta s_i^{aff}$ تبدیل می‌شود و به شکلی که در ادامه خواهیم آورد داخل معادلات وارد می‌شود. مؤلفه‌ی اصلاحی $(\Delta x^{cor},\Delta\lambda^{cor},\Delta s^{cor})$ سعی در اصلاح جهت حرکت دارد، این گام را می‌توان با استفاده از معادله های زیر یافت:

$$\begin{bmatrix}0 & A^T & I\\ A & 0 & 0\\ S & 0 & X\end{bmatrix}\begin{bmatrix}\Delta x^{cor}\\ \Delta\lambda^{cor}\\ \Delta s^{cor}\end{bmatrix}=\begin{bmatrix}0\\ 0\\ -\Delta X^{aff}\Delta S^{aff}e\end{bmatrix}$$
$$\Delta X^{aff}=diag\left(\Delta x_1^{aff},\Delta x_2^{aff},\dots,\Delta x_n^{aff}\right)$$
$$where$$
$$\Delta S^{aff}=diag(\Delta s_1^{aff},\Delta s_2^{aff},\dots,\Delta s_n^{aff}) \qquad (۳-۳۴)$$

با جمع بندی مطالب گفته شده در بالا، الگوریتم مهروترا به صورت جدول ۳-۲ که در ادامه آورده شده است ارائه می‌شود:

[1] Hadamard

**جدول ۳-۲: الگوریتم مهروترا برای برنامه ریزی خطی**

**الگوریتم پیش گو اصلاح گر مهروترا:**

فرض کنیم نقطه‌ی شروع $(x^0,\lambda^0,s^0)$ داده شده است که در آن $(x^0,s^0)>0$

**برای** $k=0,1,2,\ldots$

قرار دهید $(x,\lambda,s)=(x^k,\lambda^k,s^k)$ و معادله‌ی

$$\begin{bmatrix}0 & A^T & I\\ A & 0 & 0\\ S & 0 & X\end{bmatrix}\begin{bmatrix}\Delta x^{aff}\\ \Delta\lambda^{aff}\\ \Delta s^{aff}\end{bmatrix}=\begin{bmatrix}-r_c\\ -r_b\\ -XSe\end{bmatrix}$$

را برای $(\Delta x^{aff},\Delta\lambda^{aff},\Delta s^{aff})$; حل کنید.

حال مقادیر زیر را حساب کنید

$$\alpha_{aff}^{pri}=\arg\max\{\alpha\in[0,1]\big|x^k+\alpha\Delta x^{aff}\ge 0\}$$

$$\alpha_{aff}^{dual}=\arg\max\{\alpha\in[0,1]\big|s^k+\alpha\Delta s^{aff}\ge 0\}$$

$$\mu_{aff}=\left(x^k+\alpha_{aff}^{pri}\Delta x^{aff}\right)^T\left(s^k+\alpha_{aff}^{dual}\Delta s^{aff}\right)/n;$$

و پارامتر مرکزیت را تنظیم کنید: ; $\sigma=\left(\frac{\mu_{aff}}{\mu}\right)^3$

اکنون

$$\begin{bmatrix}0 & A^T & I\\ A & 0 & 0\\ S & 0 & X\end{bmatrix}\begin{bmatrix}\Delta x^{cc}\\ \Delta\lambda^{cc}\\ \Delta s^{cc}\end{bmatrix}=\begin{bmatrix}0\\ 0\\ \sigma\mu e-\Delta X^{aff}\Delta S^{aff}e\end{bmatrix}$$

را برای $(\Delta x^{cc},\Delta\lambda^{cc},\Delta s^{cc})$; حل کنید.

اکنون جهت جستجو و طول گام را به صورت زیر بدست آورید:

$$(\Delta x^k,\Delta\lambda^k,\Delta s^k)=(\Delta x^{aff},\Delta\lambda^{aff},\Delta s^{aff})+(\Delta x^{cc},\Delta\lambda^{cc},\Delta s^{cc});$$

$$\alpha_{max}^{pri}=\arg\max\{\alpha\ge 0|x^k+\alpha\Delta x^k\ge 0\};$$

$$\alpha_{max}^{dual}=\arg\max\{\alpha\ge 0|s^k+\alpha\Delta s^k\ge 0\}$$

حال قرار دهید:

$$\alpha_k^{pri}=\min\left(0.99*\alpha_{max}^{pri},1\right)\ and\ \alpha_k^{dual}=\min(0.99*\alpha_{max}^{dual},1);$$

مقادیر زیر را بدست آورید

$$x^{k+1}=x^k+\alpha_k^{pri}\Delta x^k;$$

$$(\lambda^{k+1},s^{k+1})=(\lambda^k,s^k)+\alpha_k^{dual}(\Delta\lambda^k,\Delta s^k);$$

**پایان(برای)**

البته شایان ذکر است که الگوریتم نوشته شده در بالا عیناً همان الگوریتمِ ارائه شده توسط مهروترا نیست بلکه الگوریتمی است که توسط بسته های نرم افزاری مثل $LIPSOL, LOQO, PCx$ پیاده سازی شده است.

## ۳-۳-۳- روش جدید ارائه شده در جهت اصلاح روش مهروترا (الگوریتم اصلاح شده‌ی مهروترا)

همان‌طور که در بخش‌های پیش از این گفته شد، روش پیش‌گو-اصلاح‌گر مهروترا به عنوان یک روش کارآمد و به عنوان زیر مجموعه ای از روش‌های پیش گو-اصلاح گر، که خود یکی از شاخه های الگوریتم‌های اولیه-دوگان هستند، ارائه شده است. از زمان ارائه‌ی این روش حدود بیست سال می گذرد و بسته های نرم افزاری که برای بهینه سازی به روش نقطه درونی نوشته شده‌اند عموماً از این الگوریتم بهره برده‌اند، بنابراین مزیت این روش امکان پیاده سازی آن در عمل است. باید به این نکته توجه کرد که بسیاری از روش‌های بهینه سازی هستند که در تئوری به خوبی عمل می‌کنند اما هنگام پیاده سازی دارای موفقیت خوبی نیستند. از این جمله الگوریتم‌ها می‌توان به "روش بیضی خاچیان"[1] اشاره نمود.

مشکل اصلی که الگوریتم اصلی مهروترا با آن مواجه است، این مسئله است که الگوریتم پیش‌گو-اصلاح‌گر مهروترا گاهی با وضعیتی روبرو می‌شود که برای آنکه در درون همسایگی خاصی تکرارها را نگاه دارد گام‌های بسیار کوچکی بر می‌دارد، که باعث می‌شود الگوریتم برای آن که به مقدار پاسخ همگرا شود، مجبور به برداشتن گام‌های بسیاری شود.

مثلاً همان‌طور که صلاحی در [۴۰] گفته است:

اگر مسئله‌ی زیر را داشته باشیم

$$\min -x_2$$
$$s.t.\ 0 \le x_1 \le 1$$
$$0 \le x_2 \le 1 + \delta x_1 \qquad (۳-۳۵)$$

اگر الگوریتم از نقطه‌ی

$$x^0 = (0.28, 0.86, 0.72, 0.1568), s^0 = (0.91, 0.5, 1, 1.5), y^0 = (-1, -1.5), \delta = 0.06$$

شروع شود، هنگامی که $(x^0, s^0) \in \mathcal{N}_\infty^-(0.5737)$ باشد، در اولین تکرار الگوریتم، گامی بسیار کوچک در گام اصلاحی بر خواهد داشت($\alpha_c = \mathcal{O}(10^{-16})$ تا سعی کند که مقادیر را در همسایگی

[1] Khachian's ellipsoid method

$\mathcal{N}_{\infty}^{-}(0.5737)$ نگاه دارد، اما در همین حال مقدارِ گام، در گام پیش بین، مقدار $\alpha_a = 0.8228$ است. بعلاوه برای چندین تکرارِ بعدی هم $\alpha_c$ کوچک می‌شود.

این مثال نشان می‌دهد که در هر همسایگی، با تغییر نقطه‌ی شروع و $\delta$ می‌توان نقطه‌ی شروع مناسبی یافت که در گام اصلاحی، الگوریتم گامی بسیار کوچک برمی‌دارد، در حالی که گام مقیاس بندی آفینی بزرگ است. بنابراین الگوریتم زیر به عنوان جبران ارائه شده است.

**قضیه ۳-۲:**

فرض کنید که برای گام فعلی $(x, y, s) \in \mathcal{N}_{\infty}^{-}(\gamma)$ و اگر فرض کنیم $(\Delta x, \Delta y, \Delta s)$ پاسخ معادله‌ی (۳-۳۷) است،

$$A\Delta x^a = 0,$$
$$A^T\Delta y^a + \Delta s^a = 0,$$
$$s\Delta x^a + x\Delta s^a = -xs \tag{۳-۳۶}$$

$$A\Delta x = 0,$$
$$A^T\Delta y + \Delta s = 0,$$
$$s\Delta x + x\Delta s = \mu e - xs - \Delta x^a \Delta s^a \tag{۳-۳۷}$$

که در آن $\mu \geq 0$ آنگاه داریم:

$$\|\Delta x \Delta s\| \leq 2^{\left(-\frac{3}{2}\right)} \left(\frac{1}{\gamma}\left(\frac{\mu}{\mu_g}\right)^2 - \left(2 - \frac{1}{2\gamma}\right)\frac{\mu}{\mu_g} + \frac{17\gamma + n}{16\gamma}\right) n\mu_g \tag{۳-۳۸}$$

برای اثبات قضیه به [۴۰] مراجعه شود.

همان‌طور که از این قضیه مشخص است مرزِ مؤلفه‌ی منفی $\Delta x^a \Delta s^a$ در مرزِ بالایی یک فاکتور $n^2$ وارد می‌کند. اکنون برای آن که مرزِ قید را در قضیه ۳-۲: کمی تیزتر کنیم، دستگاه نیوتون (۳-۳۷) را کمی عوض می‌کنیم و از نتیجه‌ی قضیه‌ی زیر زمانی استفاده می‌کنیم که بیشینه اندازه در جهتِ مقیاس بندی آفینی کمتر از یک آستانه‌ی معین باشد (برای مثال $0 < \alpha_a < 0.1$ ) و همچنین بیشینه طول گام در گام اصلاح کننده از آستانه کوچک‌تر باشد.

**قضیه ۳-۳:**

فرض کنید $\alpha_a \in (0,1]$ بیشینه اندازه‌ی طول گام، در گام پیش بین باشد. آنگاه برای هر $i \in \mathcal{I}_-$ داریم:

$$-\Delta x_i^a \Delta s_i^a \leq \frac{1}{\alpha_a}\left(\frac{1}{\alpha_a} - 1\right) x_i s_i \tag{۳-۳۹}$$

برای اثبات قضیه به [۴۰] مراجعه شود.

اکنون وقتی که $0 < \alpha_a < 0.1$ گام اصلاح کننده را با میرا کردن جمله‌ی از درجه‌ی دو در سمت راست معادله‌ی سوم (۳-۳۷) اصلاح می‌کنیم. سیستم به صورت

$$A\Delta x = 0,$$

$$A^T \Delta y + \Delta s = 0,$$

$$s\Delta x + s\Delta s = \mu e - xs - \alpha_a \Delta x^a \Delta s^a \tag{۳-۴۰}$$

تبدیل می‌شود.

**قضیه ۳-۴:**

فرض کنید که در گام فعلی $(x, y, s) \in \mathcal{N}_\infty^-(\gamma)$ و $(\Delta x, \Delta y, \Delta s)$ پاسخ (۳-۴۰) زمانی که $\mu \geq 0$ باشد. داریم:

$$\|\Delta x \Delta s\| \leq 2^{-\frac{3}{2}}\left(\frac{1}{\gamma}\left(\frac{\mu}{\mu_g}\right)^2 - \left(2 - \frac{\alpha_a}{2\gamma}\right)\left(\frac{\mu}{\mu_g}\right) + \frac{20 - 4\alpha_a + \alpha_a^2}{16}\right) n\mu_g \tag{۳-۴۱}$$

**قضیه ۳-۵:**

فرض کنید که در گام فعلی داریم $(x, y, s) \in \mathcal{N}_\infty^-(\gamma)$ و فرض کنید $(\Delta x, \Delta y, \Delta s)$ پاسخ (۳-۴۰) با $\mu = (1-\alpha_a)^3 \mu_g$ باشد و $0 < \alpha_a < 0.1$ . آنگاه:

$$\alpha_c \geq \frac{\gamma}{n} \tag{۳-۴۲}$$

زمانی که

$$\alpha_a \leq 1 - \left(\frac{2\gamma t}{1-\gamma}\right)^{\frac{1}{3}} := \alpha_1 \tag{۳-۴۳}$$

صدق کند می‌توان تضمین کرد که $\alpha_c > 0$ . اگر (۳-۴۳) برقرار نباشد داریم: $\alpha_a = \alpha_1$ و الگوریتم ادامه می‌یابد. اگر این گام‌ها منجر به طول گام کوچکی در گام اصلاح‌گر برای $\alpha_c$ شود، مقدار $\alpha_a$ را کاهش می‌دهیم.

بر پایه‌ی این تغییرات گفته شده الگوریتم جدول ۳-۳ توسط صلاحی برای مسئله‌ی برنامه ریزی خطی ارائه شده است[۴۰].

جدول ۳-۳: الگوریتم اصلاح شده‌ی مهروترا برای مسئله‌ی برنامه ریزی خطی

**الگوریتم اصلاح شده‌ی مهروترا برای برنامه ریزی خطی:**

ورودی:

یک پارامتر $\gamma \in (0, \frac{1}{4})$. یک پارامتر ایمنی برابر $\beta \in [\gamma, \frac{1}{4})$ . یک پارامتر برای دقت $\epsilon > 0$، و یک نقطه‌ی شروع $(x^0, y^0, s^0) \in \mathcal{N}_\infty^-(\gamma)$

**شروع**

تا زمانی که $x^T s \geq \epsilon$ الگوریتم زیر را تکرار کن:

**شروع**

**گام پیش گو**

معادله‌ی (۳-۳۶) را حل کنید و بیشینه طول گام $\alpha_a$ را طوری حساب کنید

که $\left(x(\alpha_a), y(\alpha_a), s(\alpha_a)\right) \in \mathcal{F}$ .(الگوریتم این گام را برنمی‌دارد)

**اگر** $(1 - \alpha_a)x(\alpha_a)^T s(\alpha_a) \leq \epsilon$ **آنگاه**

قرار دهید: $x = x(\alpha_a), s = s(\alpha_a)$ و **الگوریتم را متوقف کنید.**

**پایان**

**پایان**

**شروع**

**گام اصلاح گر**

**اگر** $\alpha_a > \alpha_1$ **آنگاه** قرار دهید $\alpha_a = \alpha_1$

**پایان**

**اگر** $\alpha_a \geq 0.1$ **آنگاه** معادله‌ی (۳-۳۷) را با $\mu$ که به صورت $\mu = \left(\frac{g_a}{g}\right)^2 \frac{g_a}{n}$ تعریف شده

حل و بیشینه طول گام $\alpha_c$ را طوری حساب کنید که $\left(x(\alpha_c), y(\alpha_c), s(\alpha_c)\right) \in \mathcal{N}_\infty^-$

**پایان**

**اگر** $\alpha_a < 0.1$ **آنگاه** معادله‌ی (۳-۴۰) را با $\mu$ که به صورت $\mu = \left(\frac{g_a}{g}\right)^2 \frac{g_a}{n}$ تعریف شده

است حل و بیشینه طول گام $\alpha_c$ را طوری حساب کنید که $\left(x(\alpha_c), y(\alpha_c), s(\alpha_c)\right) \in \mathcal{N}_\infty^-$

**پایان**

**اگر** $\alpha_a < \frac{\gamma}{\sqrt{2}n}$ **آنگاه** معادله‌ی (۳-۴۰ را با $\mu = \frac{\beta}{1-\beta}\mu_g$ حل کنید و بیشینه طول گام $\alpha_c$ را

طوری حساب کنید که $\left(x(\alpha_c), y(\alpha_c), s(\alpha_c)\right) \in \mathcal{N}_\infty^-$

**پایان**

قرار دهید $(x, y, s) = (x(\alpha_c), y(\alpha_c), s(\alpha_c))$

**پایان**

**پایان**

## ۳-۴- برنامه ریزی درجه دوم

برنامه ریزی دوم[1] به یک مسئله‌ی بهینه سازی می‌گویند که در آن تابع هزینه به صورت درجه دوم و قیدها خطی هستند. شکل کلی یک مسئله‌ی برنامه ریزی درجه دوم به صورت زیر بیان می‌شود:

$$\begin{aligned} \min_{x} \quad & q(x) = \frac{1}{2}x^T G x + x^T c \\ \text{subject to} \quad & a_i^T x = b_i, \qquad i \in \mathcal{E} \\ & a_i^T x \geq b_i, \qquad i \in \mathcal{J} \end{aligned} \tag{۳-۴۴}$$

که در توصیف بالا، $G$ یک ماتریس متقارن $nxn$ و $\mathcal{E}$ و $\mathcal{J}$ مجموعه های متناهی هستند شامل اندیس‌های قیود و $c,x,\{a_i\}$ و $i \in \mathcal{E} \cup \mathcal{J}$ بردارهایی هستند در فضای $\mathbb{R}^n$. مسائل برنامه ریزی درجه دوم را همیشه می‌توان حل نمود اما اینکه چه مقداری حل و زمان لازم است تا به نتیجه برسیم بستگی به تابع معیار و تعداد قیود دارد.

اگر ماتریس هِسین $G$ نیمه معین مثبت باشد، می‌گوییم که مسئله‌ی بالا یک مسئله‌ی برنامه ریزی درجه دوم محدب است. (برای توضیحات مبسوط درباره‌ی توابع محدب، شرایط بهینگی و اثر محدب بودن در آن به ابتدای فصل مراجعه بفرمایید)

مسائل برنامه ریزی درجه دوم محدب اکید[2] به مسائلی گویند که در آن‌ها $G$ مثبت معین است. مسائل برنامه ریزی درجه دوم نامحدب مسائلی هستند که $G$ ماتریسی نامعین است، این مسائل از آنجا که می‌توانند چندین نقطه‌ی مینیمم موضعی و نقاط ایستا داشته باشند سخت‌تر هستند. (برای توضیحات درباره‌ی نقاط مینیمم موضعی و نقاط ایستا به ابتدای فصل مراجعه بفرمایید).

برای مسائل برنامه ریزی درجه دوم محدب وقتی که $G$ مثبت نیمه معین است، شرایط KKT که در ابتدای فصل گفته شد شرایط کافی برای اینکه $x^*$ مینیمم سراسری برای تابع باشد هستند.

به طور کلی روش‌های حل مسئله‌ی برنامه ریزی درجه دوم به دسته های زیر تقسیم بندی می‌شوند:

۱- روش‌های نقطه درونی[۲۳،۲۴]
۲- روش‌های مجموعه فعال[۲۳]
۳- لاگرانژین افزوده[۲۵،۱۶]
۴- گرادیان الحاقی[۲۳]

---
[1] Quadratic Programming
[2] Strictly convex QP

۵- تصویر گرادیان[۲۳]

۶- بسط روش سیمپلکس برای مسئله برنامه ریزی درجه دوم[۲۷،۲۶]

که دو روش اول به طور کلی در مقالاتی که اخیراً ارائه شده‌اند بیشتر به کار می‌روند.

تفاوت‌های این دو روش عمده، باعث بهبود یکی بر دیگری در کاربردهای خاص آن می‌شود. به طور کلی روش‌های مجموعه فعال در مواجهه با مسائل ناشدنی[1] مناسب‌ترند. در حالی که روش‌های نقطه درونی مزیت عمده ای که دارند سرعت آن‌ها در حل مسائل است که هر چه مسئله پیچیده تر و بزرگ‌تر[2] شود این روش‌ها مزیت خود را بیشتر نشان می‌دهند چرا که دارای پیچیدگی محاسباتی از درجه چندجمله‌ای است[۲۳،۱۴].

**روش نقطه درونی**

در دهه‌ی هشتاد میلادی کشف شد که بسیاری از مسائل خطی بزرگ را با استفاده از الگوریتم‌های برنامه ریزی غیرخطی می‌توان به صورتی مناسب حل کرد. یکی از مشخصات این روش‌ها این بود که در آن‌ها باید قیدهای غیرخطی اکیداً توسط تمامی تکرارهای الگوریتم رعایت می‌شدند، بنابراین به این روش‌ها، روش‌های نقطه درونی گفته شد.

تا سال‌های اوایل دهه‌ی نود میلادی زیرمجموعه ای از روش‌های نقطه درونی که به نام روش‌های اولیه-دوگان[3] شناخته می‌شوند خود را به عنوان کارآمدترین روش‌ها در عمل و پیاده سازی مطرح کردند و توانستند به عنوان رقیبی جدی در مسائل بزرگ برای روش سیمپلکس[4] شناخته شوند. روش سیمپلکس می‌تواند در برخی مسائل خطی بسیار ناکارآمد باشد زیرا زمان مورد نیاز برای حل یک مسئله‌ی خطی در روش سیمپلکس به صورت نمایی با ابعاد مسئله افزایش می‌یابد، در حالی که در روش‌های نقطه درونی مسئله به این شکل نیست.

روش‌های نقطه درونی دارای پیچیدگی چندجمله‌ای هستند. بعلاوه یک خاصیت مشترک روش‌های نقطه درونی این است که، هر بار تکرار این روش‌ها از لحاظ محاسباتی حجیم است و در عوض هر بار تکرار پیشرفت قابل توجهی در راه پیدا کردن حل مسئله ایجاد می‌کند، بنابراین تعداد تکرارها کاهش پیدا می‌کند.

[1] Infeasible
[2] Large-Scale
[3] Primal-dual
[4] Simplex

به طور کلی دو دسته روش به عنوان زیر شاخه‌ی روش‌های نقطه درونی در نظر گرفته می‌شوند:

الف) روش‌های حامل

ب) روش‌های مسیریاب

روش‌های اولیه دوگان مسیریاب به خوبی مورد بحث و قبول واقع شده‌اند و در عمل هم پیاده سازی شده‌اند.

مسئله ای که عملاً وجود دارد این است که بعضی از روش‌های نقطه‌ی درونی با آن که در تئوری و در اثبات ریاضی دارای جواب خوبی هستند، در عمل و در برنامه‌نویسی نمی‌توانند جواب مناسبی ایجاد کنند.

### ۳-۴-۱- روش نقطه‌ی درونی برای مسائل برنامه ریزی درجه دوم

در اینجا بحث خود را محدود به این فرض که مسئله‌ی ما محدب است و قیود آن هم نامساوی هستند می‌کنیم:

$$\min_{x} \quad q(x) = \frac{1}{2}x^T G x + x^T c$$

$$subject\ to \qquad Ax \geq b \qquad (۳-۴۵)$$

که در آن ماتریس G یک ماتریس مثبت معین متقارن با ابعاد $m \times n$ است و

$$A = [a_i]_{i \in \mathcal{I}}, \qquad b = [b_i]_{i \in \mathcal{I}}, \quad \mathcal{I} = \{1,2,\dots,m\} \qquad (۳-۴۶)$$

اکنون با این شرایط مجدداً اقدام به نوشتن شرایط KKT می‌کنیم. با توجه به نکات در نظر گرفته شده در بالا برای مسئله‌ی برنامه ریزی درجه دوم محدب داریم:

$$Gx - A^T\lambda + c = 0$$

$$Ax - b \geq 0$$

$$(Ax - b)_i \lambda_i = 0, \quad i = 1,2,\dots,m$$

$$\lambda \geq 0 \qquad (۳-۴۷)$$

اکنون در معادلات بالا، برداری به معادلات نامساوی می‌افزاییم که به آن بردار کمکی[1] می‌گوییم. شرطی که برای آن در نظر می‌گیریم این است که $y \geq 0$

$$
\begin{gathered}
Gx - A^T\lambda + c = 0 \\
Ax - y - b = 0 \\
y_i\lambda_i = 0, \quad i = 1,2,\dots,m \\
(y,\lambda) \geq 0
\end{gathered}
\tag{۳-۴۸}
$$

از آنجا که ماتریس هِسین $G$ را مثبت معین فرض کرده‌ایم، شرایط KKT برای آن که نقاط کمینه را بیابند نه تنها شرایط لازم هستند بلکه شرایط کافی نیز هستند (برای اطلاع بیشتر از قواعد اینکه مثبت معین بودن ماتریس هسین چه شرایطی را برای مسائل برنامه ریزی درجه دوم بوجود می‌آورد و این که این شرایط-محدب بودن-چگونه باعث می‌شود که شرایط KKT شرایط کافی باشند به ابتدای فصل مراجعه بفرمایید).
اکنون با فرض اینکه در تکرار فعلی در نقطه‌ی $(x,\lambda,y)$ هستیم و داریم $(y,\lambda) > 0$ معیار زیر را به عنوان معیار مکمل تعریف می‌کنیم:

$$\mu = \frac{y^T\lambda}{m} \tag{۳-۴۹}$$

شرایط KKT برای روش‌های اولیه-دوگان به صورت زیر داده می‌شود:

$$
\begin{gathered}
F(x,y,\lambda;\sigma\mu) = \begin{bmatrix} Gx - A^T\lambda + c \\ Ax - y - b \\ \mathcal{Y}\Lambda e - \sigma\mu e \end{bmatrix} = 0 \\
\mathcal{Y} = diag(y_1, y_2, \dots, y_m), \\
\Lambda = diag(\lambda_1, \lambda_2, \dots, \lambda_m), \\
e = (1,1,\dots,1)^T, \qquad \sigma \in [0,1]
\end{gathered}
\tag{۳-۵۰}
$$

حل معادلات بالا برای تمامی مقادیر مثبت $\sigma,\mu$ مسیر مرکزی را می‌دهد، این مسیر هنگامی که $\sigma\mu$ به سمت صفر میل کند جواب مسئله‌ی درجه دوم را می‌دهد.

[1] Slack vector

با اعمال روش نیوتون :

$$\begin{bmatrix} G & 0 & -A^T \\ A & -I & 0 \\ 0 & \Lambda & \mathcal{Y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p \\ -\Lambda \mathcal{Y} e + \sigma \mu e \end{bmatrix} \qquad (۵۱\text{-}۳)$$

که در آن

$$r_d = Gx - A^T \lambda + c, \quad r_p = Ax - y - b \qquad (۵۲\text{-}۳)$$

برای بدست آوردن تکرار بعدی داریم:

$$(x^+, y^+, \lambda^+) = (x, y, \lambda) + \alpha(\Delta x, \Delta y, \Delta \lambda) \qquad (۵۳\text{-}۳)$$

بیشتر عملیاتِ محاسباتی در روش نقاط درونی بدست آوردن جواب معادله‌ی

$$\begin{bmatrix} G & 0 & -A^T \\ A & -I & 0 \\ 0 & \Lambda & \mathcal{Y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p \\ -\Lambda \mathcal{Y} e + \sigma \mu e \end{bmatrix} \qquad (۵۴\text{-}۳)$$

است. از آنجا که در این حالت ماتریس هِسین را در معادلات داریم، محاسبات نسبت به حالت برنامه ریزی خطی می‌تواند بسیار حجیم تر باشد. بنابراین اینکه برای حل مسئله‌ی تکرارشونده، پیش از آن که وارد حل مسئله شویم تمهیداتی بیندیشیم به نظر ضروری می‌رسد.

### ۳-۴-۲- توسعه‌ی الگوریتم اصلاح شده‌ی مهروترا برای حل مسئله‌ی برنامه ریزی درجه دوم

محبوب‌ترین روش برای حل مسئله‌ی برنامه ریزی درجه دومِ محدب، بر پایه‌ی روشِ پیش گو-اصلاح گرِ مهروترا است که در اصل برای مسائل برنامه ریزی خطی ارائه شده است و در این نوشتار هم به آن اشاره شده است. در بخش پیشین هم الگوریتم اصلاح شده‌ی آن که به نام الگوریتم اصلاح شده‌ی مهروترا معرفی شده است، آورده شده است.

اکنون توسعه‌ی الگوریتم اصلاح شده‌ی مهروترا برای برنامه ریزی درجه دوم را ارائه خواهیم کرد. همان طور که می‌دانیم برای مسئله‌ی برنامه ریزی درجه دوم داریم:

$$\min q(x) = \frac{1}{2} x^T G x + x^T x$$

$$subject\ to\ Ax \geq b \qquad (۵۵\text{-}۳)$$

$$Gx - A^T\lambda + c = 0$$
$$Ax - b \geq 0$$
$$(Ax-b)_i\lambda_i = 0 \quad i = 1,2,\dots,m$$
$$\lambda \geq 0 \tag{۳-۵۶}$$

با معرفی ضرایب لاگرانژ داریم:

$$Gx - A^T\lambda + c = 0$$
$$Ax - b - y = 0$$
$$y_i'\lambda_i = 0 \quad i = 1,2,\dots,m$$
$$(y,\lambda) \geq 0 \tag{۳-۵۷}$$

از آنجا که فرض کرده‌ایم ماتریس $G$ مثبت نیمه معین است، شرایط KKT نه تنها شرایط لازم بلکه شرایط کافی برای بهینگی است و بنابراین می‌توان با حل (۳-۵۷) به جواب برای مسئله‌ی (۳-۵۵) رسید.

همانند برنامه ریزی خطی، در اینجا هم برای رسیدن به زیر مجموعه ای از روش‌های اولیه-دوگان شرایط KKT تغییر یافته که به صورت معادله‌ی (۳-۵۸) است را حل می‌کنیم.

$$F(x,y,\lambda;\sigma\mu) = \begin{bmatrix} Gx - A^T\lambda + c \\ Ax - y - b \\ \mathcal{Y}\Lambda e - \sigma\mu e \end{bmatrix} = 0$$
$$\mathcal{Y} = diag(y_1, y_2, \dots, y_m), \quad \Lambda = diag(\lambda_1, \lambda_2, \dots, \lambda_m) \tag{۳-۵۸}$$

با اعمال روش نیوتون به سیستم خطی زیر برای سیستمِ غیر شدنی می‌رسیم:

$$\begin{bmatrix} G & 0 & -A^T \\ A & -I & 0 \\ 0 & \Lambda & \mathcal{Y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p \\ -\Lambda\mathcal{Y}e + \sigma\mu e \end{bmatrix} \tag{۳-۵۹}$$

که در آن:

$$r_d = Gx - A^T\lambda + c, \quad r_p = Ax - y - b \tag{۳-۶۰}$$

هم چنین در گام اصلاح گر بر طبق روش‌های اولیه-دوگان باید داشته باشیم:

$$\begin{bmatrix} G & 0 & -A^T \\ A & -I & 0 \\ 0 & \Lambda & \mathcal{Y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p \\ -\Lambda\mathcal{Y}e - \Delta\Lambda^{aff}\Delta\mathcal{Y}^{aff} + \sigma\mu e \end{bmatrix} \tag{۳-۶۱}$$

حال با استفاده از مفاهیم گفته شده توسط صلاحی [۴۰] داریم:

$$\mathcal{F} = \{(x,\lambda,y) \in \mathcal{R}^n \times \mathcal{R}^m \times \mathcal{R}^n | Ax - y = b, Gx - A^T\lambda + c = 0, (y,\lambda) > 0 \tag{۳-۶۲}$$

$$\mu = \left(\frac{g_a}{g}\right)^2 \left(\frac{g_a}{n}\right) \tag{۳-۶۳}$$

همچنین

$$x(\alpha) = x + \alpha\Delta x \tag{۳-۶۴}$$

$$\lambda(\alpha) = \lambda + \alpha\Delta\lambda \tag{۳-۶۵}$$

$$y(\alpha) = y + \alpha\Delta y \tag{۳-۶۶}$$

حال در نظر بگیرید:

$$\alpha_a \le 1 - \left(\frac{2\gamma t}{1-\gamma}\right)^{\frac{1}{3}} := \alpha_1 \tag{۳-۶۷}$$

آنگاه:

**قضیه ۳-۶:**

فرض کنید که برای گام فعلی $(x,\lambda,y) \in \mathcal{N}_\infty^-(\gamma)$ و اگر فرض کنیم $(\Delta x, \Delta\lambda, \Delta y)$ جواب معادله‌ی (۳-۶۹) است،

$$\begin{aligned} G\Delta x^a - A^T\Delta\lambda &= -r_d, \\ \Lambda\,\Delta y^a + \mathcal{Y}\Delta\lambda^a &= -\Lambda\mathcal{Y}e + \sigma\mu e, \\ A\Delta x^a - I\Delta y^a &= -r_p \end{aligned} \tag{۳-۶۸}$$

$$\begin{aligned} G\Delta x^c - A^T\Delta\lambda^c &= -r_d, \\ A\Delta x^c - I\Delta y^c &= -r_p \\ \Lambda\,\Delta y^c + \mathcal{Y}\Delta\lambda^c &= -\Lambda\mathcal{Y}e + \sigma\mu e - \Delta\Lambda^{aff}\Delta\mathcal{Y}^{aff}e, \end{aligned} \tag{۳-۶۹}$$

که در آن $\mu \ge 0$ آنگاه داریم:

$$\|\Delta x \Delta y\| \leq 2^{\left(-\frac{3}{2}\right)}\left(\frac{1}{\gamma}\left(\frac{\mu}{\mu_g}\right)^2-\left(2-\frac{1}{2\gamma}\right)\frac{\mu}{\mu_g}+\frac{17\gamma+n}{16\gamma}\right)n\mu_g \tag{۳-۷۰}$$

اکنون قضیه‌ی زیر را ببینید.

**قضیه ۳-۷:**

فرض کنید $\alpha_a \in (0,1]$ بیشینه اندازه‌ی طول گام، در گام پیش بین باشد. آنگاه برای هر $i \in \mathcal{I}_-$ داریم:

$$-\Delta x_i^a \Delta y_i^a \leq \frac{1}{\alpha_a}\left(\frac{1}{\alpha_a}-1\right)x_i y_i \tag{۳-۷۱}$$

اکنون وقتی که $0 < \alpha_a < 0.1$ گام اصلاح کننده را با میرا کردن جمله‌ی از درجه‌ی دو در سمت راست معادله‌ی سوم (۳-۳۷) اصلاح می‌کنیم. سیستم به صورت

$$G\Delta x - A^T\Delta\lambda = -r_d,$$

$$A\Delta x - I\Delta y = -r_p$$

$$\Lambda\,\Delta y + \mathcal{Y}\Delta\lambda = -\Lambda\mathcal{Y}e + \sigma\mu e - \alpha_a\Delta\Lambda^{aff}\Delta\mathcal{Y}^{aff}e, \tag{۳-۷۲}$$

در می‌آید.

**قضیه ۳-۸:**

فرض کنید که در گام فعلی $(x,\lambda,y) \in \mathcal{N}_\infty^-(\gamma)$ و $(\Delta x,\Delta\lambda,\Delta y)$ جواب (۳-۷۲) زمانی که $\mu \geq 0$ باشد. داریم:

$$\|\Delta x \Delta y\| \leq 2^{-\frac{3}{2}}\left(\frac{1}{\gamma}\left(\frac{\mu}{\mu_g}\right)^2-\left(2-\frac{\alpha_a}{2\gamma}\right)\left(\frac{\mu}{\mu_g}\right)+\frac{20-4\alpha_a+\alpha_a^2}{16}\right)n\mu_g \tag{۳-۷۳}$$

**قضیه ۳-۹:**

فرض کنید که در گام فعلی داریم $(x,\lambda,y) \in \mathcal{N}_\infty^-(\gamma)$ و فرض کنید $(\Delta x,\Delta\lambda,\Delta y)$ جواب (۳-۴۰) با $\mu = (1-\alpha_a)^3\mu_g$ باشد و $0 < \alpha_a < 0.1$ . آنگاه:

$$\alpha_c \geq \frac{\gamma}{n} \tag{۷۴-۳}$$

زمانی که

$$\alpha_a \leq 1 - \left(\frac{2\gamma t}{1-\gamma}\right)^{\frac{1}{3}} \coloneqq \alpha_1 \tag{۷۵-۳}$$

صدق کند می‌توان تضمین کرد که $\alpha_c > 0$ . اگر(۳-۴۳) برقرار نباشد داریم: $\alpha_a = \alpha_1$ و الگوریتم ادامه می‌یابد. اگر این گام‌ها منجر به طول گام کوچکی در گام اصلاح گر برای $\alpha_c$ شود، مقدار $\alpha_a$ را کاهش می‌دهیم.

اکنون الگوریتم اصلاح شده‌ی مهروترا برای مسئله‌ی برنامه ریزی درجه دوم به صورت زیر بیان می‌شود:

جدول ۳-۴: توسعه‌ی الگوریتم اصلاح شده‌ی مهروترا برای مسئله‌ی برنامه ریزی درجه دوم

---

**الگوریتم اصلاح شده‌ی مهروترا برای برنامه ریزی درجه دوم:**

---

ورودی:

یک پارامتر $\gamma \in (0,\frac{1}{4})$. یک پارامتر ایمنی برابر $\beta \in [\gamma,\frac{1}{4})$ . یک پارامتر برای دقت $\epsilon > 0$، و یک نقطه‌ی شروع $(x^0, y^0, \lambda^0) \in \mathcal{N}_\infty^-(\gamma)$

**شروع**

تا زمانی که $y^T\lambda \geq \epsilon$ الگوریتم زیر را تکرار کن:

**شروع**

**گام پیش گو**

معادله‌ی (۳-۵۹) را با $\mu = 0$ حل کنید و بیشینه طول گام $\alpha_a$ را طوری حساب کنید که $\big(x(\alpha_a), \lambda(\alpha_a), y(\alpha_a)\big) \in \mathcal{F}$ .(الگوریتم این گام را برنمی‌دارد)

**اگر** $(1-\alpha_a)\lambda(\alpha_a)^T y(\alpha_a) \leq \epsilon$ **آنگاه**

قرار دهید: $\lambda = \lambda(\alpha_a), y = y(\alpha_a)$ و **الگوریتم را متوقف کنید.**

**پایان**

**پایان**

**شروع**

**گام اصلاح گر**

**اگر** $\alpha_a > \alpha_1$ **آنگاه** قرار دهید $\alpha_a = \alpha_1$

**پایان**

**اگر** $\alpha_a \geq 0.1$ **آنگاه** معادله‌ی (۳-۳۷) را با $\mu$ که به صورت $\mu = \left(\frac{g_a}{g}\right)^2 \frac{g_a}{n}$ تعریف شده حل و بیشینه طول گام $\alpha_c$ را طوری حساب کنید که $\big(x(\alpha_c), \lambda(\alpha_c), y(\alpha_c)\big) \in \mathcal{N}_\infty^-$

**پایان**

**اگر** $\alpha_a < 0.1$ **آنگاه** معادله‌ی (۳-۷۲) را با $\mu$ که به صورت $\mu = \left(\frac{g_a}{g}\right)^2 \frac{g_a}{n}$ تعریف شده است حل و بیشینه طول گام $\alpha_c$ را طوری حساب کنید که $\big(x(\alpha_c), \lambda(\alpha_c), y(\alpha_c)\big) \in \mathcal{N}_\infty^-$

**پایان**

**اگر** $\alpha_a < \frac{\gamma}{\sqrt{2n}}$ **آنگاه** معادله‌ی (۳-۷۲) را با $\mu = \frac{\beta}{1-\beta}\mu_g$ حل کنید و بیشینه طول گام $\alpha_c$ را طوری حساب کنید که $\big(x(\alpha_c), \lambda(\alpha_c), y(\alpha_c)\big) \in \mathcal{N}_\infty^-$

**پایان**

قرار دهید $(x, \lambda, y) = (x(\alpha_c), \lambda(\alpha_c), y(\alpha_c))$

**پایان**

**پایان**

---

## ۳-۵- بررسی الگوریتم‌های مورد استفاده در جعبه ابزار بهینه سازی (ویرایش ۵٫۰) نرم افزار متلب

جعبه ابزار فعلی نرم افزار متلب ۷٫۱۰٫۰ از دستور qpdantz در جعبه ابزار کنترل پیش بین (ویرایش ۳٫۲)[۱] برای محاسبه‌ی مسئله‌ی بهینه سازی کنترل پیش بین استفاده می‌کند. این دستور از الگوریتم دانتزیگ-ولف[۲] که از جمله روش‌های مجموعه‌ی فعال در روش‌های نقطه درونی است، استفاده می‌کند. همچنین نرم افزار متلب علاوه بر جعبه ابزار کنترل پیش بین یک جعبه ابزار تخصصی هم به نام جعبه ابزار بهینه سازی دارد.

در این جعبه ابزار خاص (ویرایش ۵٫۰)، اگر بخواهیم یک مسئله‌ی برنامه ریزی خطی را حل کنیم می‌توانیم از دستور linprog استفاده کنیم. این دستور برای مسائل با ابعاد وسیع از بسته نرم افزاری $LIPSOL$[۳] که گونه ای است از الگوریتم مهروترا است استفاده می‌کند که همان‌طور که پیش از این گفته شد یک نوع الگوریتم پیش گو-اصلاح گر اولیه-دوگان نقطه درونی[۴] است.

این دستور برای مسائل با ابعاد متوسط از روش تصویری[۵] استفاده می‌کند و گونه ای از روش سیمپلکس را برای مسئله حل می‌کند. این الگوریتم در ابتدا با حل یک مسئله‌ی برنامه ریزی خطی دیگر یک راه حل شدنی[۶] اولیه برای مسئله پیدا می‌کند.

اگر مسائل بخواهند از نقطه نظری مورد بررسی قرار گیرند که با نرم افزار متلب به صورت برنامه ریزی درجه دوم حل شوند باید در نرم افزار از دستور"Quadprog" استفاده کرد. این دستور در حل مسائل ابعاد وسیع از روش زیر فضای قابل اعتماد[۷] که بر پایه‌ی روش نیوتون انعکاس دهنده‌ی میانی[۸] است استفاده می‌کند.

برای مسائل با ابعاد متوسط این دستور از روش تصویری استفاده می‌کند که مرجع مناسب برای آن مرجع [۵۱] است.

[1] Model predictive control toolbox
[2] Dantzig-Wolfe's algorithm
[3] Linear Interior-Point solver
[4] Predictor-corrector primal-dual interior-point method
[5] Projection method
[6] Feasible
[7] Subspace trust-region method
[8] Interior-reflective Newton method

# فصل ۴– کاربرد روش اصلاح شده‌ی مهروترا در حل مسئله‌ی کنترل پیش بین

## ۴–۱– پیش گفتار

همان‌طور که در فصل‌های پیشین گفته شد، مسئله‌ی کنترل پیش بین مقید در نهایت نیازمند حل یک مسئله‌ی بهینه سازی درجه دوم است. در فصل دوم نیز اشاره شد که با توجه به آنکه مدلِ سیستم در کنترل پیش بین چگونه توصیف گردد (با استفاده از فضای حالت و یا با استفاده از تابع تبدیل) ماهیت بردارِ متغیرها در مسئله‌ی بهینه سازی تغییر خواهد کرد.

بنابراین برای آنکه بخواهیم یک الگوریتمِ بهینه سازی را به مسئله‌ی کنترل پیش بین اعمال کنیم، در ابتدا ضروری است که مشخص گردد از چه ابزاری برای توصیف سیستم استفاده شده است. باید توجه داشت که در صورتی که از مدل فضای حالت استفاده گردد، همیشه قیدی به صورت زیر در مسئله وجود خواهد داشت:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du \qquad (۱-۴)$$

در صورتی که اگر از مدل تابع تبدیل استفاده کنیم، این قید وجود ندارد و معادلاتِ سیستم در روابطی که منجر به مسئله‌ی بهینه سازی شده‌اند نهفته است.

مسئله‌ی بهینه سازی درجه دوم که به صورت معادله‌ی (۴-۲) تعریف می‌شود را در نظر بگیرید:

$$\begin{aligned} \min_{x} \qquad & q(x) = \frac{1}{2}x^TGx + x^Tc \\ \textit{subject to} \qquad & a_i^Tx = b_i, \qquad i \in \mathcal{E} \\ & a_i^Tx \ge b_i, \qquad i \in \mathcal{J} \end{aligned} \qquad (۲-۴)$$

در این مسئله نیز همان‌طور که در فصل دوم اشاره شد، بردار متغیرها ($x$) می‌تواند از عناصر متفاوتی تشکیل شده باشد (دقت کنید که این $x$ ها متغیرهای مسئله‌ی بهینه سازی هستند که در بالا تعریف شده است و نباید آن‌را با حالت‌های سیستم که در مدل فضای حالت عموماً با نماد $x$ نمایش داده می‌شود اشتباه گرفت). این عناصر می‌توانند شامل بردار ورودی‌های کنترلی ($u$ها) سیستم که هدف تعیین آن‌ها

تا افق کنترلی مدنظر است، تغییرات ورودی‌های کنترلی در هر گام مسئله ($\Delta u$ ها)، یا شامل هر دو بردار ورودی‌های کنترلی پیش بینی شده و تغییرات ورودی‌های کنترلی پیش بینی شده برای هر گام مسئله باشد. همچنین در صورتی که از مدلِ فضای حالت استفاده کنیم می‌توان خودِ حالت‌های سیستم را هم در بردار متغیرهای مسئله‌ی بهینه سازی وارد کرد.

## ۴-۲- شبیه سازی و نتایج

در این پایان نامه یک مسئله‌ی GPC مورد بررسی قرار گرفته است، همان‌طور که می‌دانیم در مسئله‌ی GPC سیستم با استفاده از مدل تابع تبدیل توصیف می‌شود، همچنین همان‌طور که در فصل دوم دیدیم در نهایت در GPC به مسئله‌ی بهینه سازی زیر می‌رسیم:

$$
\begin{gathered}
J = \frac{1}{2}u^T H u + b^T u + f_0 \\
H = 2(G^T G + \lambda I) \\
b^T = 2(f-w)^T (f-w) \\
f_0 = (f-w)^T (f-w) \\
u = \begin{bmatrix} \Delta u(t) \\ \Delta u(t+1) \\ \vdots \\ \Delta u(t+N-1) \end{bmatrix} \\
\Delta = 1 - z^{-1}
\end{gathered}
\tag{۳-۴}
$$

حال باید برای مسئله‌ی بهینه سازی بدست آمده‌ی فوق الگوریتم مناسب انتخاب و اعمال گردد. در فصل سوم شرح داده شد که روش‌های نقطه‌ی درونی با توجه به آنکه دارای پیچیدگی محاسباتی از درجه چندجمله‌ای هستند، گزینه‌ی مناسبی برای مسائل ابعاد وسیع می‌باشند. هر چه ابعاد مسئله‌ی بهینه سازی افزایش یابد، سرعت پاسخ دهی آن‌ها نسبت به سایر روش‌ها (مثل سیمپلکس و روش‌های مبتنی بر مجموعه‌ی فعال) بیشتر می‌شود. بنابراین برای هر مسئله ای حدی وجود دارد که هنگامی که ابعاد مسئله‌ی بهینه سازی بزرگ‌تر از آن مقدار می‌شود، سرعت پاسخ دهی الگوریتم‌های نقطه درونی از الگوریتم‌های دیگر (که دارای پیچیدگی محاسباتی از درجه نمایی هستند) بیشتر می‌گردد.

بنابراین در این پایان نامه از آنجا که هدف ارائه‌ی روش‌های سریع بهینه سازی برای مسئله‌ی کنترل پیش بین است، روش‌های نقطه درونی انتخاب شده است. همچنین از آن‌جا که هدف پیاده سازی موفق بوده است، از میان الگوریتم‌ها و شاخه های گوناگون روش‌های نقطه درونی، الگوریتمی انتخاب شده است که در عمل هم موفق‌ترین الگوریتم پیاده سازی شده توسط بسته های نرم افزاری بهینه سازی بوده است.

همان‌طور که در فصل سوم بیان گشت با روش نقطه درونی پیش گو-اصلاح گر اولیه –دوگان مهروترا این هدف حاصل می‌گردد. (برای جزئیات این روش به فصل سوم مراجعه شود).

در ادامه، الگوریتم ارائه شده در مرجع [۴۰] که به تفصیل در فصل سوم شرح داده شد به مسئله‌ی کنترل پیش بین (۴-۳) اعمال می‌گردد. از آنجا که این الگوریتم برای مسئله‌ی برنامه ریزی خطی ارائه گشته است، ابتدا این الگوریتم برای مسئله‌ی برنامه ریزی درجه دوم توسعه داده شده و سپس به مسئله کنترل پیش بین اعمال گشته است.

الگوریتم انتخاب شده‌ی پیش‌گو-اصلاح‌گر مهروترا از مرجع [۲۳] آورده شده است که برای مسئله‌ی برنامه ریزی درجه دوم ارائه شده است.

### ۴-۲-۱- اعمال الگوریتم مهروترا به مسئله‌ی کنترل پیش بین تعمیم یافته

مسئله‌ی GPC زیر (برگرفته از [۴۳]) را در نظر بگیرید:

$$(1 + az^{-1})y(t) = (b_0 + b_1 z^{-1})u(t-1) + \frac{e(t)}{\Delta} \qquad (۴\text{-}۴)$$

در این مثال تأخیر $d$ برابر صفر در نظر گرفته شده است و چندجمله‌ای نویز $C(z^{-1})$ برابر با ۱ فرض شده است.

الگوریتم GPC را با مقادیر عددی $a = -0.8,\ b_0 = 0.4,\ b1 = 0.6$ در نظر می‌گیریم. در اینجا اگر فرض کنیم که $N_2 = N_u = 3$ است، ابتدا مقادیر پیش بینی شده برای خروجی فرآیند برای افق مورد نظر محاسبه می‌شود و به فرم معادلات (۲-۲۳) نوشته می‌شود، سپس قانون کنترل با استفاده از حل مسئله‌ی بهینه سازی بدست می‌آید.

چندجمله‌ای‌های پیش بین $E_j(z^{-1})\,, F_j(z^{-1})$ از $j = 1$ تا $j = 3$ را می‌توان با حل معادله‌ی دیوفانتین با در نظر گرفتن:

$$\tilde{A}(z^{-1}) = A(z^{-1})(1 - z^{-1}) = 1 - 1.8z^{-1} + 0.8z^{-2} \qquad (۵\text{-}۴)$$

بدست آورد. برای این مسئله‌ی خاص که افق خیلی گسترده نیست، می‌توان چندجمله‌ای ها را با تقسیم کردن ۱ بر $\tilde{A}(z^{-1})$ بدست آورد.

$$E_1(z^{-1}) = 1 \qquad F_1(z^{-1}) = 1.8 - 0.8z^{-1}$$

$$E_2 = 1 + 1.8z^{-1} F_2 = 2.44 - 1.44z^{-1}$$

$$E_3 = 1 + 1.8z^{-1} + 2.44z^{-2} F_3 = 2.952 - 1.952z^{-1} \qquad (۶\text{-}۴)$$

باید توجه داشت که برای برنامه‌نویسی از این روش نمی‌توان استفاده کرد زیرا برنامه‌ای که توسط نویسنده‌ی این پایان نامه توسعه یافته است، با این قابلیت نوشته شده است که بتواند مسئله‌ی کنترل پیش بین را با هر افقی اجرا و شبیه سازی کند. بنابراین در برنامه‌ی کامپیوتری از روش زیر استفاده شده است:

```
A_tilde(1)=A(1);                    % trying to make system's equations
for j=2:size_A
A_tilde(j)=A(j)-A(j-1);
end
A_tilde(size_A+1)=-A(size_A);
F=zeros(P,size_A+1);
F(1,:)=[-A_tilde(1,2:size_A+1) 0];
               for j=1:P-1          %trying to make system's equations
                                    (solvingsylvester's equations)
fori=1:size_A
F(j+1,i)=F(j,i+1)-A_tilde(i+1)*F(j,1);
end
end
% Now we should make the matrix E
E=zeros(P,P);
E(1,1)=1;
for j=1:P-1                         %trying to make system's equations
                                    (solvingsylvester's equations)
E(j+1,:)=E(j,:);
E(j+1,j+1)=F(j,1);
end
```

این بخش از آن جهت دارای اهمیت است که معادلات دیوفانتین به صورت بازگشتی حل می‌شود.

اکنون برای مسئله‌ی عددی فرض شده، با داشتن $E_1, E_2, E_3, F_1, F_2, F_3$ چندجمله‌ای $B(z^{-1})$ به صورت زیر بدست می‌آید:

$$B(z^{-1}) = 0.4 + 0.6z^{-1} \tag{۷-۴}$$

مقادیر $G_i(z^{-1})$ بدست می‌آید:

$$G_1 = 0.4 + 0.6z^{-1}$$

$$G_2 = 0.4 + 1.32z^{-1} + 1.08z^{-2}$$

$$G_3 = 0.4 + 1.32z^{-1} + 2.056z^{-2} + 1.464z^{-3} \tag{۸-۴}$$

بنابراین خروجی‌های پیش بینی شده به صورت زیر بدست می‌آیند:

$$\begin{bmatrix}\hat{y}(t+1|t)\\ \hat{y}(t+2|t)\\ \hat{y}(t+3|t)\end{bmatrix} = \begin{bmatrix}0.4 & 0 & 0\\ 1.32 & 0.4 & 0\\ 2.056 & 1.32 & 0.4\end{bmatrix}\begin{bmatrix}\Delta u(t)\\ \Delta u(t+1)\\ \Delta u(t+2)\end{bmatrix} + \begin{bmatrix}0.6\Delta u(t-1) + 1.8y(t) - 0.8y(t-1)\\ 1.08\Delta u(t-1) + 2.44y(t) - 1.44y(t-1)\\ 1.464\Delta u(t-1) + 2.952y(t) - 1.952y(t-1)\end{bmatrix} \tag{۹-۴}$$

که در آن:

$$f = \begin{bmatrix}0.6\Delta u(t-1) + 1.8y(t) - 0.8y(t-1)\\ 1.08\Delta u(t-1) + 2.44y(t) - 1.44y(t-1)\\ 1.464\Delta u(t-1) + 2.952y(t) - 1.952y(t-1)\end{bmatrix} \tag{۱۰-۴}$$

از آنجا که داریم:

$$w = [w(t+d+1) \quad w(t+d+2) \quad \dots \quad w(t+d+N)]^T \tag{۱۱-۴}$$

باید مرجع مطلوب[1] ($w$) از پیش برای سیستم مشخص باشد تا سیستم پیش بینی خود را برای رسیدن به آن هدف انجام دهد. بدین منظور بر اساس آنچه که در مرجع این مثال نیز در نظر گرفته شده است، سیستم برای ۱۲ ثانیه کار با در نظر گرفتن مرجع مطلوب برابر با ۱ واحد شبیه سازی و پیاده سازی می‌شود. بر این اساس داریم

$$w = [1 \quad \dots \quad 1]^T \tag{۱۲-۴}$$

و

$$G = \begin{bmatrix}0.4 & 0 & 0\\ 1.32 & 0.4 & 0\\ 2.056 & 1.32 & 0.4\end{bmatrix} \tag{۱۳-۴}$$

با داشتن ماتریس $G$ و با انتخاب وزن $\lambda$ برابر با ۰/۸ ماتریس $H$ - ماتریس هِسین مسئله‌ی بهینه‌سازی- با عبارت زیر بدست می‌آید:

$$H = 2(G^TG + \lambda I) \tag{۱۴-۴}$$

همچنین همان‌طور که پیش از این مشاهده گشت، مقدار عددی برای ماتریس $f$ نیز بدست می‌آید. بنابراین می‌توان مقدار ماتریس $b$ مسئله‌ی بهینه سازی را به صورت زیر بدست آورد:

[1] Desired Reference

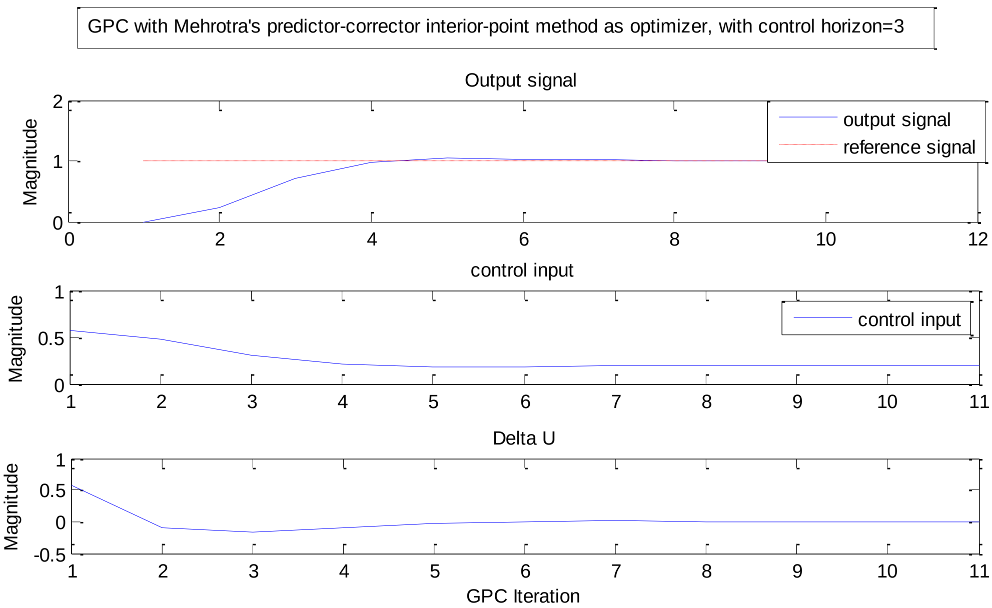
$$b^T = 2(f-w)^T G \qquad (۱۵-۴)$$

همان‌طور که در معادله‌ی (۴-۱۰) قابل مشاهده است، سطر $i$ ام ماتریس $f$ وابسته به پارامترهای $\Delta u(t-1)$، $y(t)$ و $y(t-1)$ است. بنابراین در هر گام برای پیش بینی خروجی‌ها و ورودی‌های کنترلی آینده نیاز به خروجی فعلی سیستم (در همان لحظه)، خروجی لحظه‌ی قبل و تغییرات قانون کنترل در لحظه قبل داریم. باید توجه داشت، اگرچه در مسئله‌ی بهینه سازی GPC عبارت دیگری هم به صورت $f_0$ داریم که برابرست با:

$$f_0 = (f-w)^T (f-w) \qquad (۱۶-۴)$$

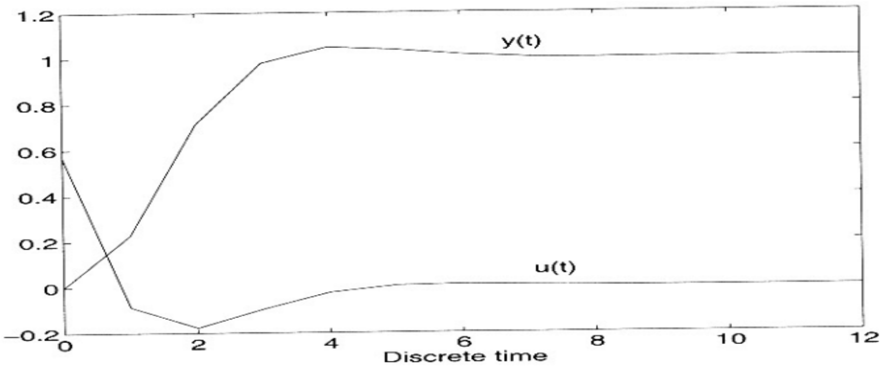
اما از آنجایی که این عبارت در صورتِ تابعی که می‌خواهیم آن‌را کمینه کنیم تنها به صورت یک ماتریس عددی اضافه شده است (وابستگی به پارامتر متغیر مسئله‌ی بهینه سازی یعنی $u$ ندارد) مقدار آن تأثیری در $u$ بدست آمده برای سیستم ندارد و تنها در مقدار عددی تابع بهینه سازی $J$ موثر است. بنابراین در حل مسئله‌ی بهینه سازی می‌توان این عبارت ثابت را حذف نمود.

حال با اجرای نرم افزار نوشته شده به صورت تکرارشونده، نمودار زیر بدست می‌آید.




**شکل ۴-۱: کنترل پیش بین با افق کنترل ۳، برای ۱۲ گام تکرار و با اجرای بهینه سازی توسط الگوریتم مهروترا**

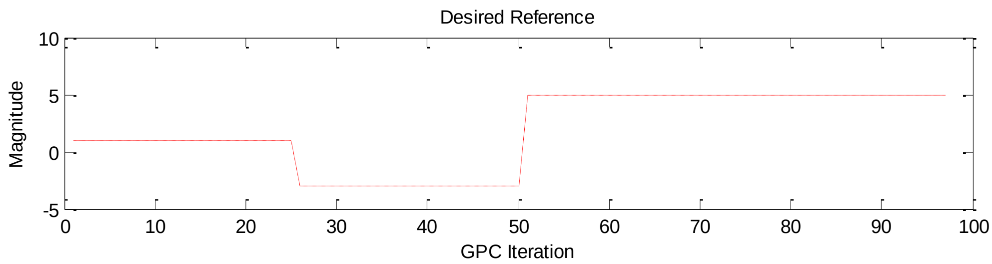
از طرفی مرجع این مثال نیز، نمودار خروجی را به صورت شکل۴-۲ رسم کرده است (به نمودار ورودی$u$ توجه کنید. دقت نمایید که متغیر $u$ در شکل مرجع برابر است با متغیر $\Delta u$ در برنامه‌ی نوشته شده برای این پایان نامه).




**شکل۴-۲: شکل خروجی و سیگنال کنترلِ مثال کنترل پیش بین در مرجع[۴۳]**

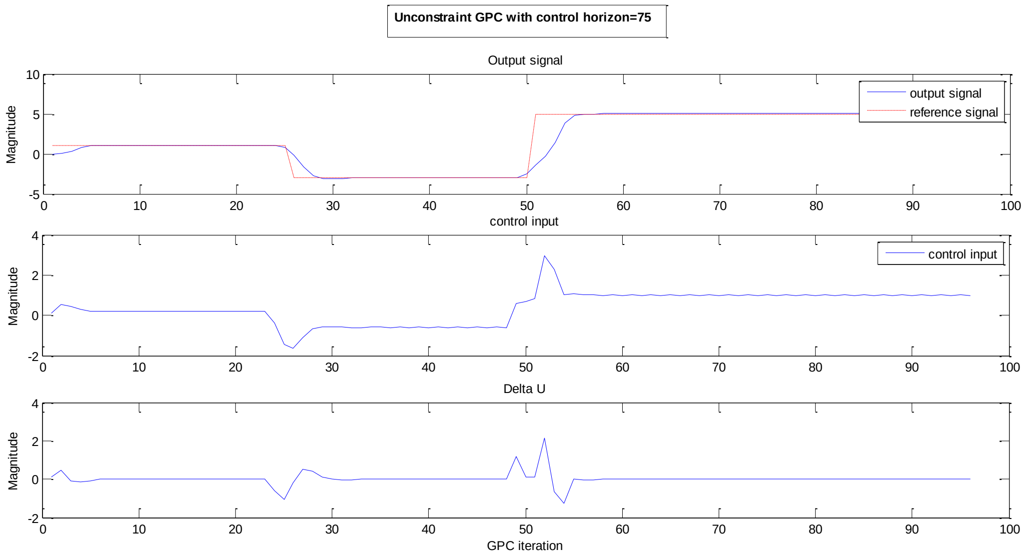
حال فرض نمایید که می‌خواهیم ابعاد مسئله‌ی بهینه سازی را افزایش دهیم. در این مسئله برای افزایش ابعاد مسئله‌ی بهینه سازی ناگزیر به افزایشِ افق پیش بینی هستیم. اگر مسئله را با همان مقادیر پیشین برای مخرج و صورت تابع تبدیل سیستم در نظر بگیریم (سیستم را تغییر ندهیم) و تنها افق کنترلی را تا ۷۵ تکرار زیاد کنیم (به این معنی که در هر بار تکرار الگوریتم کنترل پیش بین، باید تا ۷۵ گام جلوتر ورودی‌های کنترلی مطلوب محاسبه شود و خروجی حاصل از آن‌ها هم در نظر گرفته شود تا به مرجع مطلوب برسد) و اگر این کار را برای ۱۰۰ گام مسئله‌ی کنترل پیش بین انجام دهیم، نتایج زیر بدست می‌آید. در نظر داشته باشید از آنجا که الگوریتم برای پیش بینی ۷۵ گام جلوتر نیاز به داشتن مرجع مطلوب تا ۷۵ گام بعد را دارد و همچنین می‌خواهیم تا گام صدم این الگوریتم تکرار شود، در اجرای گام صدم کنترل پیش بین، الگوریتم قانون کنترلی پیش بینی شده‌ی خود را تا گام ۱۷۵ بدست می‌آورد. پس باید مرجع مطلوب برای آن تا گام ۱۷۵ تعیین شده باشد.

ورودی مرجع را برابر پله‌هایی به صورت شکل۴-۳ در نظر می‌گیریم:




**شکل۴-۳: ورودی مرجع اعمال شده به سیستم**

اگر به ازای این سیگنال مرجع، مسئله‌ی کنترل پیش بین را اجرا نماییم، خروجی به صورت زیر خواهد بود:




**شکل ۴-۴: کنترل پیش بین تعمیم یافته با افق ۷۵ با استفاده از بهینه سازی به روش مهروترا**

حال می‌خواهیم اثرات وارد کردن قیود را در نتایج بررسی کنیم، همان‌طور که گفته شد یکی از مزیت‌های عمده‌ی کنترل پیش بین که باعث کاربرد وسیع آن در صنایع بخصوص در کنترل فرآیندها شده است، توانایی کنترل پیش بین در بیان صریح قیود است (مقدمه و فصل دوم را ببینید)، بنابراین با وارد کردن قیود به بررسی اثرات وارد کردن قید بر نتایج بدست آمده می‌پردازیم.

از آنجا که مسئله‌ی بهینه سازی دارای بردار متغیرها به صورت زیر است:

$$u = \begin{bmatrix} \Delta u(t) \\ \Delta u(t+1) \\ \vdots \\ \Delta u(t+N-1) \end{bmatrix} \qquad (۱۷-۴)$$

اگر قیدی در مسئله‌ی بهینه سازی به صورت ساده (یعنی به صورت $u_{min} < u < u_{max}$) بیان شود، در واقع در مسئله‌ی کنترل پیش بین تغییراتِ سیگنال کنترلی در هر گام اجرای الگوریتم کنترل پیش بین را محدود بین دو مقدار کرده‌ایم. حال فرض کنید داریم:

$$J = \frac{1}{2} u^T H u + b^T u + f_0$$

$$s.t. \qquad u_{min} < u < u_{max} \qquad (۱۸-۴)$$

با همان اعداد مسئله‌ی کنترل پیش بین تعمیم یافته که به فرم معادله‌ی(۴-۱۹) است، مسئله را حل می‌کنیم.

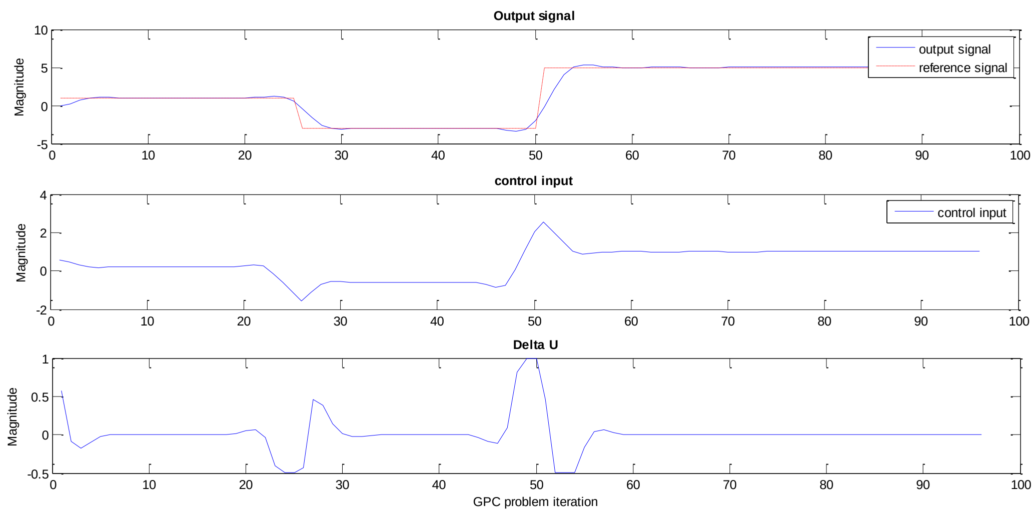
$$\tilde{A}(z^{-1}) = A(z^{-1})(1 - z^{-1}) = 1 - 1.8z^{-1} + 0.8z^{-2}$$

$$B(z^{-1}) = \ 0.4 + 0.6z^{-1}$$

$$G = \begin{bmatrix} 0.4 & 0 & 0 \\ 1.32 & 0.4 & 0 \\ 2.056 & 1.32 & 0.4 \end{bmatrix}$$

$$f = \begin{bmatrix} 0.6\Delta u(t-1) + 1.8y(t) - 0.8y(t-1) \\ 1.08\Delta u(t-1) + 2.44y(t) - 1.44y(t-1) \\ 1.464\Delta u(t-1) + 2.952y(t) - 1.952y(t-1) \end{bmatrix} \tag{۴-۱۹}$$

اکنون یک مسئله‌ی کنترلِ پیش بین مقید داریم. اگر مقدار $\Delta u$ را به صورت $-0.5 < \Delta u < 1$ محدود کنیم، منحنی خروجی سیستم ($y$)، سیگنالِ کنترلی $u$ و تغییراتِ سیگنالِ کنترلی $\Delta u$ به صورت شکل۴-۵ برای مسئله با افق کنترل ۷۵ و برای یک‌صد بار تکرار گام‌های کنترل پیش بین بدست می‌آید:

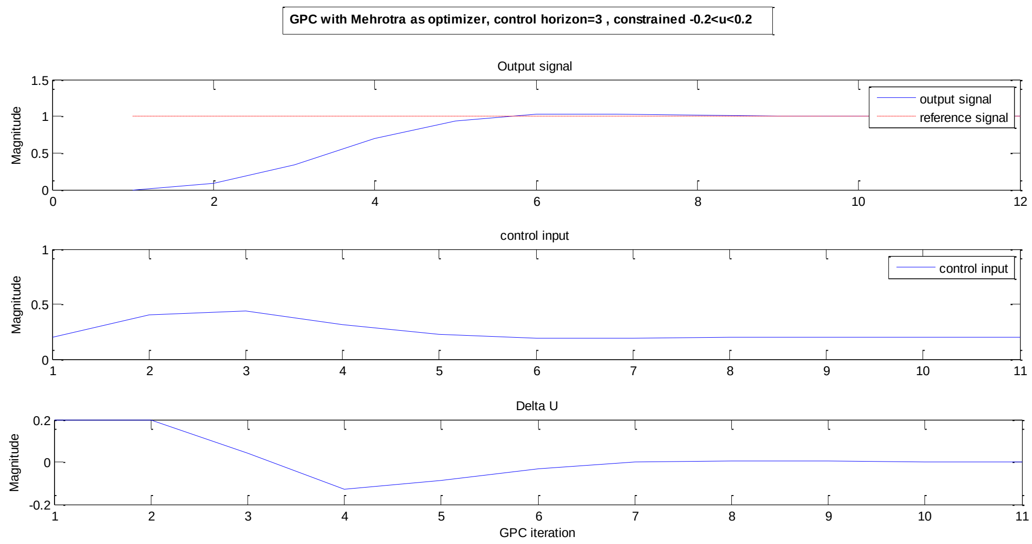



**شکل۴-۵: کنترل پیش بین تعمیم یافته‌ی مقید با افق کنترلی ۷۵ با بهینه ساز مهروترا**

به طور مشابه وقتی که افق کنترلی برابر ۳ است و سیستم برای ۱۲ گام مورد بررسی قرار می‌گیرد، اگر قید زیر نیز برقرار باشد:

$$-0.2 < u < 0.2 \tag{۴-۲۰}$$

پاسخ سیستم به صورت زیر حاصل می‌گردد:




**شکل۴-۶: کنترل پیش بین مقید با افق کنترلی ۳ برای ۱۲ بار تکرار با بهینه ساز مهروترا**

بنابراین می‌بینیم که ذکر قیود به صورت صریح، می‌تواند به راحتی در کنترل پیش بین انجام شود.

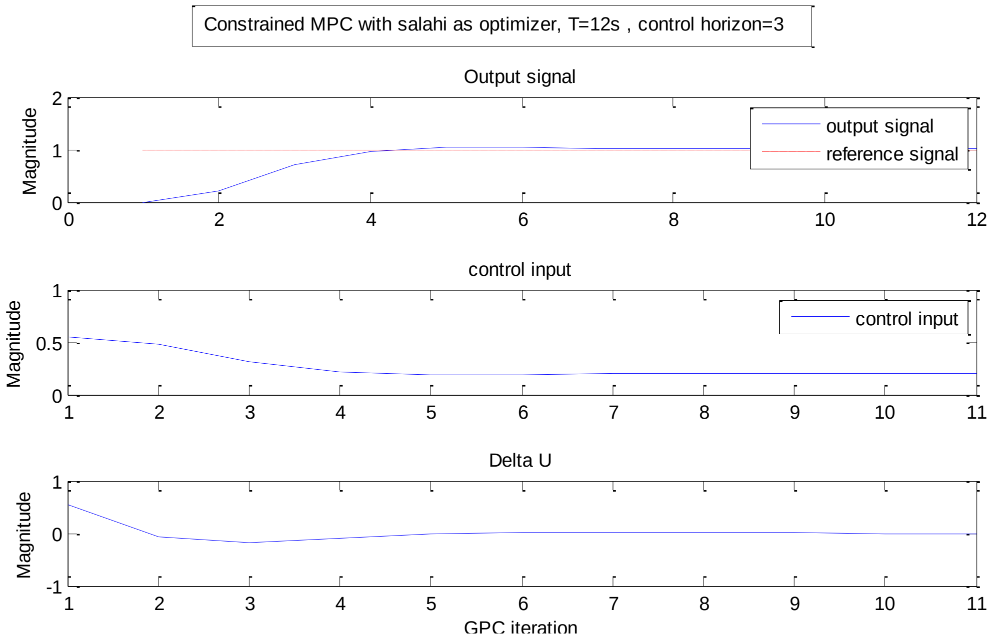
## ۴-۲-۲- اعمال الگوریتم اصلاح شده‌ی مهروترا به مسائل مختلف کنترل پیش بین تعمیم یافته

همان‌طور که در فصل سوم اشاره شد، به تازگی الگوریتمی ارائه شده است که گونه‌ی خاصی از الگوریتم مهروترا است (در واقع تغییراتی در الگوریتم مهروترا داده است و آن‌را اصلاح کرده است). این الگوریتم که در سال ۲۰۰۶ ارائه شده است، برای مسئله‌ی بهینه سازی خطی ارائه شده است. در اینجا با توجه به اینکه برای استفاده در مسئله‌ی کنترل پیش بین نیاز به حل مسئله‌ی بهینه سازی درجه دوم داریم، باید این الگوریتم برای مسئله‌ی برنامه ریزی درجه دوم توسعه یابد، که در فصل سوم توسعه آن به صورت درجه دوم شرح داده شد. در ادامه به بررسی شبیه سازی‌های انجام شده و مقایسه نتایج حاصل شده با دستور نرم افزار متلب می‌پردازیم.

### ۴-۲-۲-۱- بررسی صحت عملکرد برنامه‌ی کامپیوتری الگوریتم اصلاح شده‌ی مهروترا

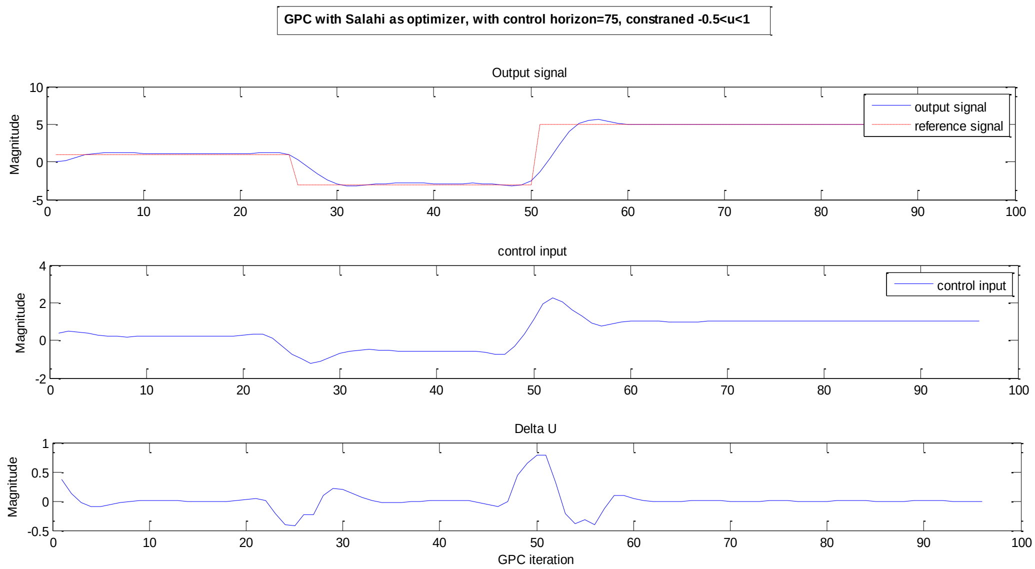
اگر برنامه کامپیوتری لازم برای الگوریتم اصلاح شده‌ی مهروترا را بنویسیم و برای مثال عددی مطرح شده در ابتدای فصل آن‌را اجرا کنیم، نتایجی را که در ادامه آورده شده است حاصل می‌گردد.

در حالت اول فرض نمایید که مسئله با قید $-0.5 < u < 1$ محدود شده است و افق کنترلی برابر ۳ و اجرا در ۱۵ تکرار در نظر گرفته شده است. بر این اساس خواهیم داشت:




**شکل ۴-۷: شبیه سازی جواب سیستم به ازای ۱۲ گام اجرا با افق کنترلی ۳ و بهینه ساز اصلاح شده‌ی مهروترا**

در حالت دوم فرض نمایید که مسئله با قید $-0.5 < u < 1$ محدود گشته است و افق کنترلی برابر ۷۵ و اجرا در ۱۰۰ تکرار در نظر گرفته شده است. بر این اساس پاسخ زیر حاصل می‌گردد:




**شکل ۴-۸: جواب سیستم به ازای ۱۰۰ گام اجرا با افق کنترلی ۷۵ و بهینه ساز اصلاح شده‌ی مهروترا**

با مقایسه‌ی نتایج حاصل شده در این بند با بند گذشته می‌توان دید که الگوریتم اصلاح شده‌ی مهروترا نیز به همان صورت و پاسخ‌های یکسانی تولید می‌کند. اکنون که از صحت عملکرد برنامه‌ی نوشته شده برای الگوریتمِ اصلاح شده‌ی مهروترا اطمینان حاصل کردیم، می‌توان به مقایسه‌ی سرعت پاسخ‌دهی آن با دستورِ نرم افزار متلب پرداخت.

### ۴-۲-۲-۲- مقایسه‌ی الگوریتم‌های اصلاح شده‌ی مهروترا با دستور متلب از لحاظ سرعت پاسخ دهی

در این بخش به مقایسه‌ی مدّت زمان لازم برای آنکه نرم افزار جواب‌های تولید شده در بالا را برای دو الگوریتم مختلف تولید کند خواهیم پرداخت. یکی الگوریتم اصلاح شده‌ی مهروترا و دیگری الگوریتمِ دستورِ نرم افزار متلب برای حل مسائل بهینه سازیِ درجه دوم[1] که برای مسائل ابعاد وسیع از الگوریتمِ زیرفضای قابل اعتماد، که بر پایه‌ی روشِ نیوتونِ انعکاس دهنده‌ی میانی است استفاده می‌کند ([۵۰] را ببینید) و برای مسائل ابعاد متوسط از یک نوع الگوریتم مجموعه فعال (که همچنین گونه ای از روش‌هایِ منعکس کننده نیز می‌باشد) استفاده می‌کند[۵۱] (برای توضیحات بیشتر به توضیحات نرم افزار متلب مراجعه شود[۵۲]). بدین منظور باید توجه داشت که این محاسبات با نرم افزار متلب ورژن ۷٫۱۰٫۰و بر روی دستگاه کامپیوتر با مشخصات زیر انجام شده است که در آزمایشگاه کنترل پیشرفته در دانشگاه صنعتی خواجه نصیرالدین طوسی در زمان انجام این پایان نامه موجود بوده است:

CPU: Intel® CoreI™2 Duo ,
E7300 @ 2.66GHz,
3.50 GB of RAM

در جدول زیر نتایج مقایسه‌ی شبیه سازی‌های انجام شده در بالا را برای هر دو الگوریتم می‌بینید:

**جدول۴-۱: سرعت پاسخ‌دهی الگوریتم‌های اصلاح شده‌ی مهروترا و دستورِ متلب برای سیستم نمونه‌ی شماره‌ی یک**

| سیستم با `A=[1 -0.8];B=[0.4 0.6];` | | |
|---|---|---|
| افق کنترل مسئله‌ی کنترل پیش بین | زمان (ثانیه) | |
| | Salahi’s variant of Mehrotra’s PC IPM | quadprog |
| ۳ گام جلوتر | ۰٫۵۴۷۲۶۸ | ۰٫۹۲۴۲۸۰ |
| ۱۰ گام جلوتر | ۰٫۶۶۸۴۸۲ | ۰٫۹۶۳۵۴۵ |
| ۲۰ گام جلوتر | ۰٫۹۵۴۳۴۵ | ۱٫۰۲۱۷۳۵ |

نکته‌ی دیگر قابل بحث این است که اگرچه در اینجا سرعت انجام الگوریتم مورد بررسی قرار گرفته است، اما از آنجا که دستورات متلب با میان بُرهای گوناگون نوشته شده است و از آنجا که در نهایت برای

[1] Quadprog()

حل مسئله‌ی بهینه سازی از زبان برنامه نویسی $C$ استفاده شده است برای مقایسه‌ی عادلانه باید هر دو الگوریتم توسط یک نفر و در یک محیط برنامه نویسی نوشته شوند.

اکنون باید دید که اگر سیستم تغییر کند چه اتفاقی خواهد افتاد. به این منظور قطب‌ها و صفرهای سیستم را تغییر می‌دهیم. همان‌طور که در جدول‌ها و در شکل‌ها مشخص است، با تغییر سیستم هم تغییری در نتیجه بوجود نمی‌آید. در جداول زیر مقادیر مختلفی برای سیستم انتخاب شده است، قطب‌ها در بعضی حالات پایدارند و در برخی از حالات ناپایدار می‌باشند، همچنین برای صفرهای سیستم، مقادیر مختلفی از بزرگ (در حدود ۱۵۰) تا کوچک در نظر گرفته شده است.

**جدول۴-۲:سرعت پاسخ‌دهی الگوریتم‌های اصلاح شده‌ی مهروترا و دستورِ متلب برای سیستم نمونه‌ی شماره‌ی دو**

| سیستم با ; A=[1 -1 -0.8]; B=[0.4 0.6] | | |
|---|---|---|
| افق کنترل مسئله‌ی کنترل پیش بین | زمان (ثانیه) | |
| | Salahi’s variant of Mehrotra’s PC IPM | quadprog |
| ۳ گام جلوتر | ۰,۵۳۵۸۰۷ | ۰,۹۴۲۵۱۸ |
| ۱۰ گام جلوتر | ۰,۶۶۸۴۸۲ | ۰,۹۶۳۵۴۵ |
| ۲۰ گام جلوتر | ۰,۸۵۷۹۲۴ | ۰,۹۷۸۵۷۳ |

**جدول۴-۳:سرعت پاسخ‌دهی الگوریتم‌های اصلاح شده‌ی مهروترا و دستورِ متلب برای سیستم نمونه‌ی شماره‌ی سه**

| سیستم با A=[1 -1 -0.8]; B=[0.04 -6] | | |
|---|---|---|
| افق کنترل مسئله‌ی کنترل پیش بین | زمان (ثانیه) | |
| | Salahi’s variant of Mehrotra’s PC IPM | Quadprog |
| ۳ گام جلوتر | ۰,۵۳۵۸۰۷ | ۰,۸۹۶۱۷۱ |
| ۱۰ گام جلوتر | ۰,۷۸۸۶۱۳ | ۰,۹۲۳۴۶۳ |
| ۲۰ گام جلوتر | ۰,۸۰۵۱۸۳ | ۰,۸۴۵۶۵۳ |

جدول۴-۴: سرعت پاسخ‌دهی الگوریتم‌های اصلاح شده‌ی مهروترا و دستورِ متلب برای سیستم نمونه‌ی شماره‌ی چهار

| سیستم با A=[1 -1 0.675]; B=[0.04 -6] | | |
|---|---|---|
| افق کنترل مسئله‌ی کنترل پیش بین | زمان (ثانیه) | |
| | Salahi’s variant of Mehrotra’s PC IPM | Quadprog |
| ۳ گام جلوتر | ۰,۷۴۸۹۵۳ | ۰,۹۱۲۴۲۸ |
| ۱۰ گام جلوتر | ۰,۸۸۴۵۶۸ | ۰,۹۲۳۲۳۰ |
| ۲۰ گام جلوتر | ۰,۸۱۵۲۸۱ | ۰,۹۴۹۰۳۴ |

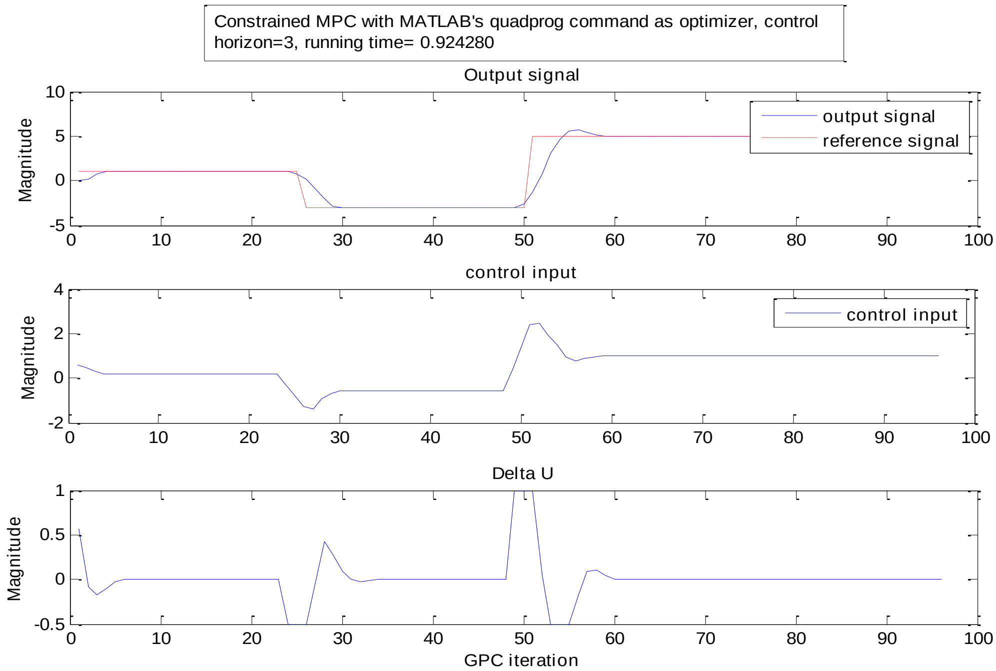
### ۴-۲-۲-۳- نتیجه‌گیری از نتایج بدست آمده

همان‌طور که در جدول‌های جدول۴-۱ تا جدول۴-۴ قابل مشاهده است، برنامه‌ی متلب نوشته شده برای الگوریتم اصلاح شده‌ی مهروترا سریع‌تر است. با توجه به نتایج جداول مشخص است که هر چه ابعاد مسئله بهینه سازی بزرگ‌تر گردد، زمان لازم برای پاسخگویی در مورد دستور نرم افزار متلب با نرخ کمتری در حال افزایش است. به این معنی که با افزایش ابعاد ماتریس هِسین به تدریج برنامه‌ای که برای الگوریتم اصلاح شده‌ی مهروترا نوشته شده است، کارایی خود را در مقابل برنامه‌ی دستور متلب از دست می‌دهد. این پدیده را می‌توان این گونه توجیه کرد که الگوریتم نرم افزار متلب پیش از حل مسئله بهینه سازی از دستورات زیادی به منظور اسپارس کردن مسئله استفاده کرده است، به عبارتی با روش‌های هوشمند از حجم مسئله بهینه سازی کاسته است. حال آن که برای الگوریتم اصلاح شده‌ی مهروترا، اگرچه الگوریتم سرعت بیشتری دارد اما از آنجا که برنامه‌ای که در متلب برای آن نوشته شده است به صورت آماتوری نوشته شده است و ضرب‌های ماتریسی عیناً انجام شده است، مزیت استفاده از برنامه نویسی بهینه را دارا نیست و این امر سبب می‌شود که با افزایش ابعاد مسئله قابلیت رقابت خود را از دست دهد. راه حلی که در ادامه این پایان نامه برای رفع این نقص پیشنهاد می‌گردد، بازنویسی برنامه‌ی کامپیوتری نرم افزار متلب و مقایسه‌ی آن با برنامه‌ی کامپیوتری نوشته شده برای الگوریتمِ اصلاح شده‌ی مهروترا است. در شکل‌های زیر پاسخ‌هایی که در جدول‌های بالا زمانِ پاسخ‌دهی برای آن‌ها ذکر شده بود، شبیه سازی و رفتار خروجی حاصل شده مقایسه شده است.

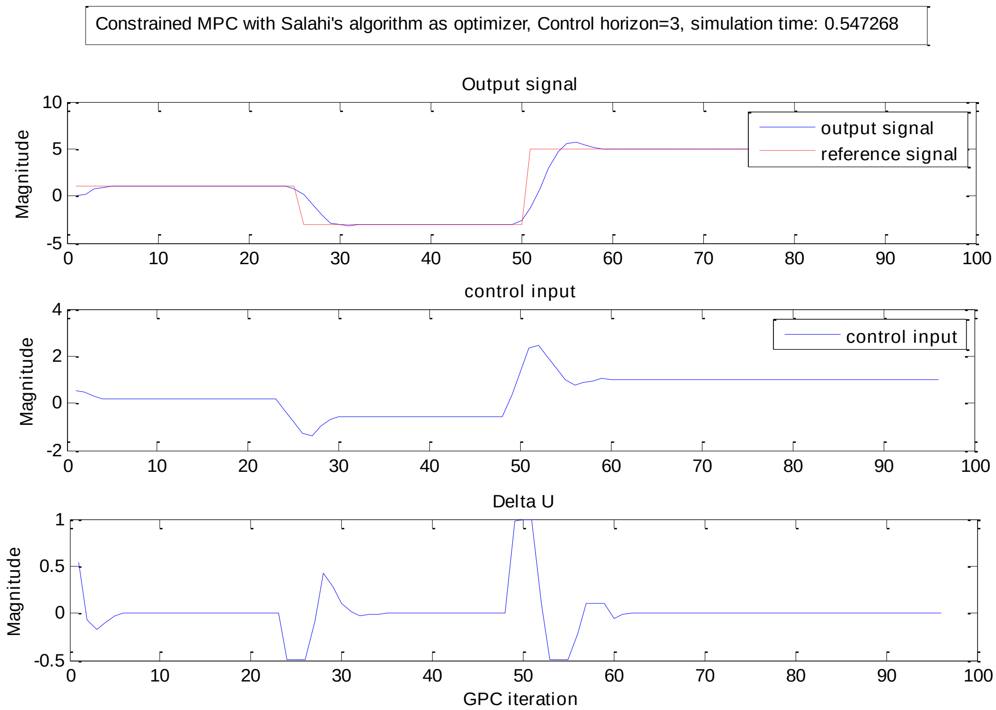
## ۴–۲–۳– مقایسه از روی شکل

### ۴-۲-۳-۱- سیستم به صورت A=[1 -0.8]; B=[0.4 0 .6];

افق کنترل برابر ۳:

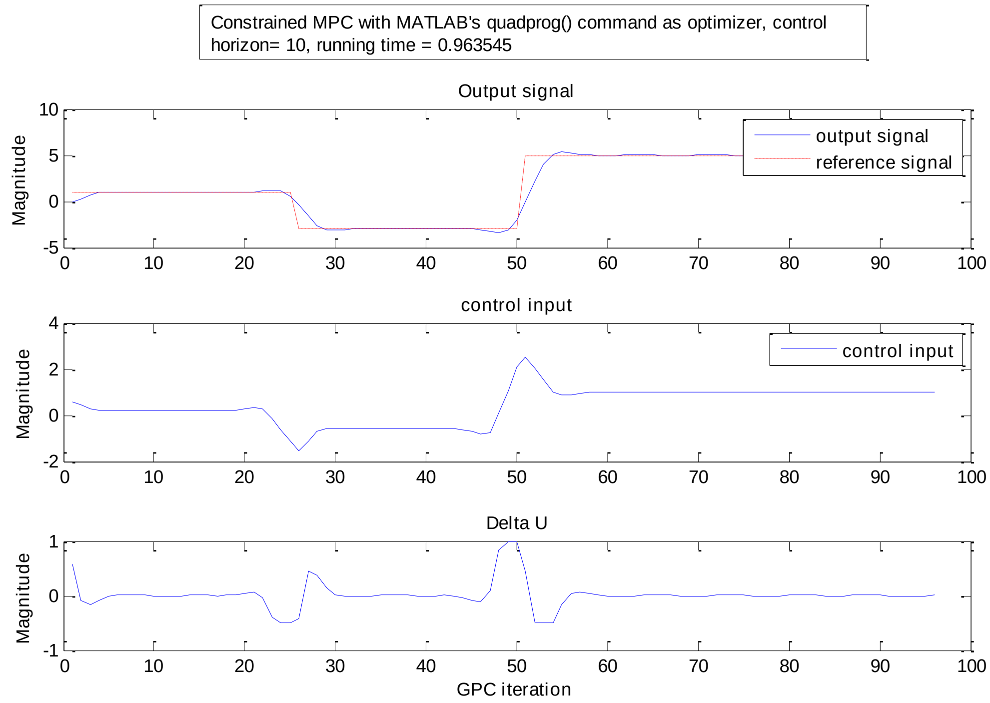



شکل ۴-۹: سیستم A= [1 -0.8]; B=[0.4 0 .6]; با افق کنترل ۳ با بهینه ساز متلب

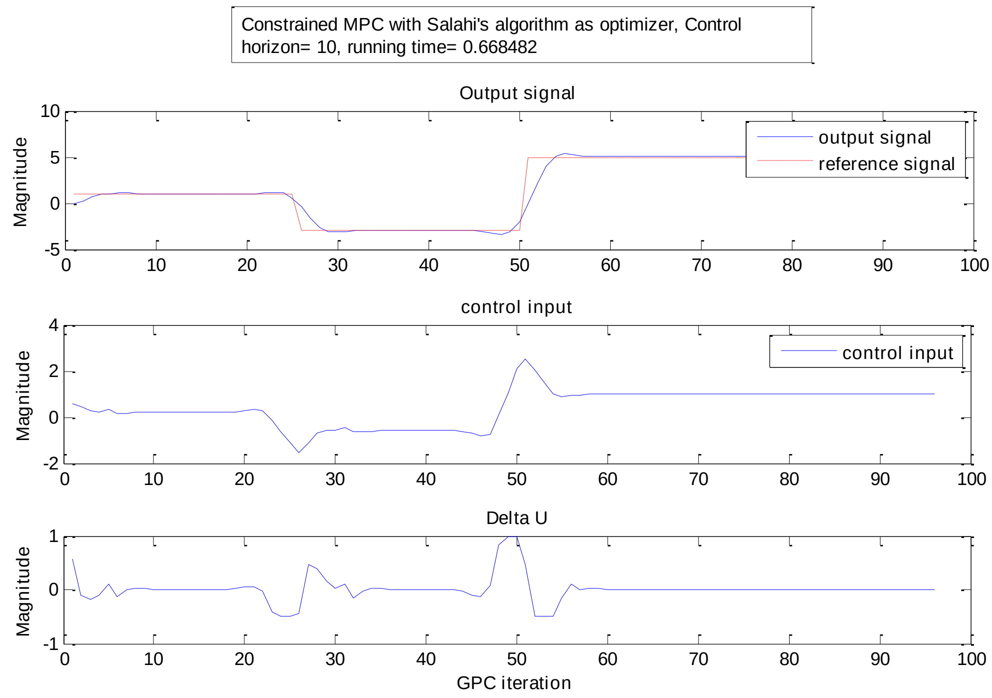



شکل ۴-۱۰: سیستم A= [1 -0.8]; B=[0.4 0 .6]; با افق کنترل ۳ با بهینه ساز اصلاح شده‌ی مهروترا

### افق کنترل برابر ۱۰:

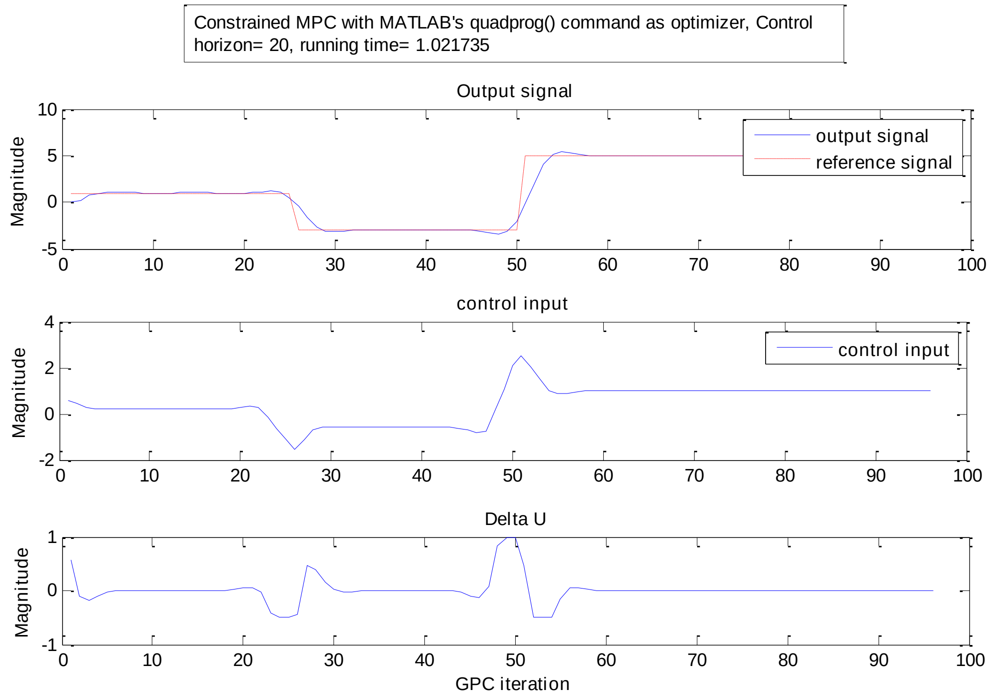



شکل ۴-۱۱: سیستم ;[0.4   0 .6]=B   ;[0.8-   1] =A با افق کنترل ۱۰ با بهینه ساز متلب

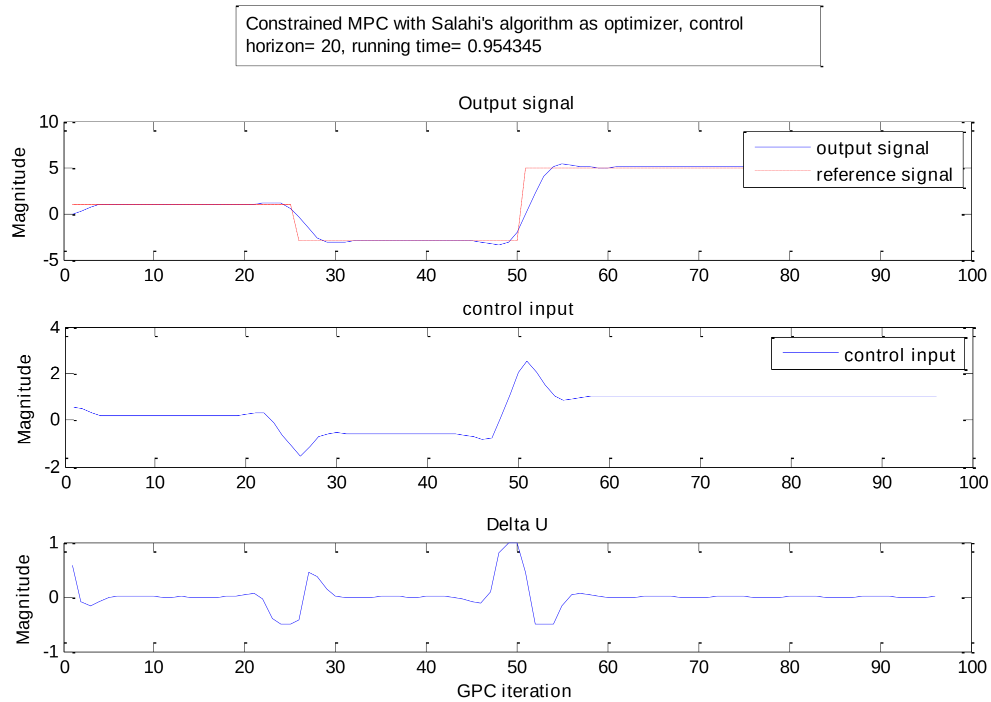



شکل ۴-۱۲: سیستم ;[0.4   0 .6]=B   ;[0.8-   1] =A با افق کنترل ۱۰ با بهینه ساز اصلاح شده‌ی مهروترا

## افق کنترل برابر ۲۰:



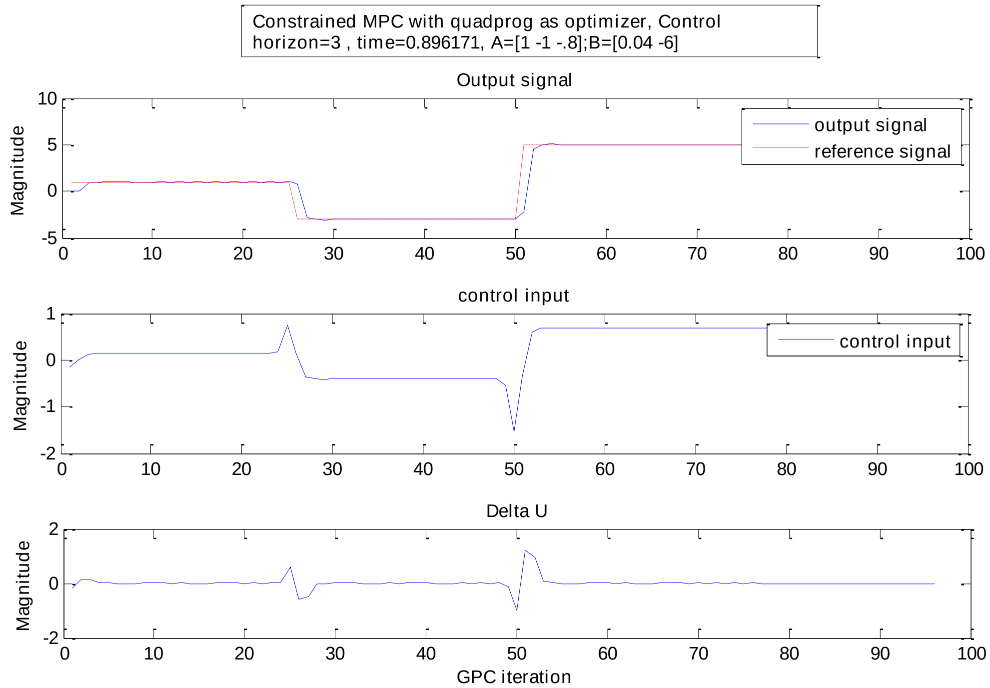

شکل ۴-۱۳: سیستم A= [1 -0.8]; B=[0.4 0.6]; با افق کنترل ۲۰ با بهینه ساز متلب



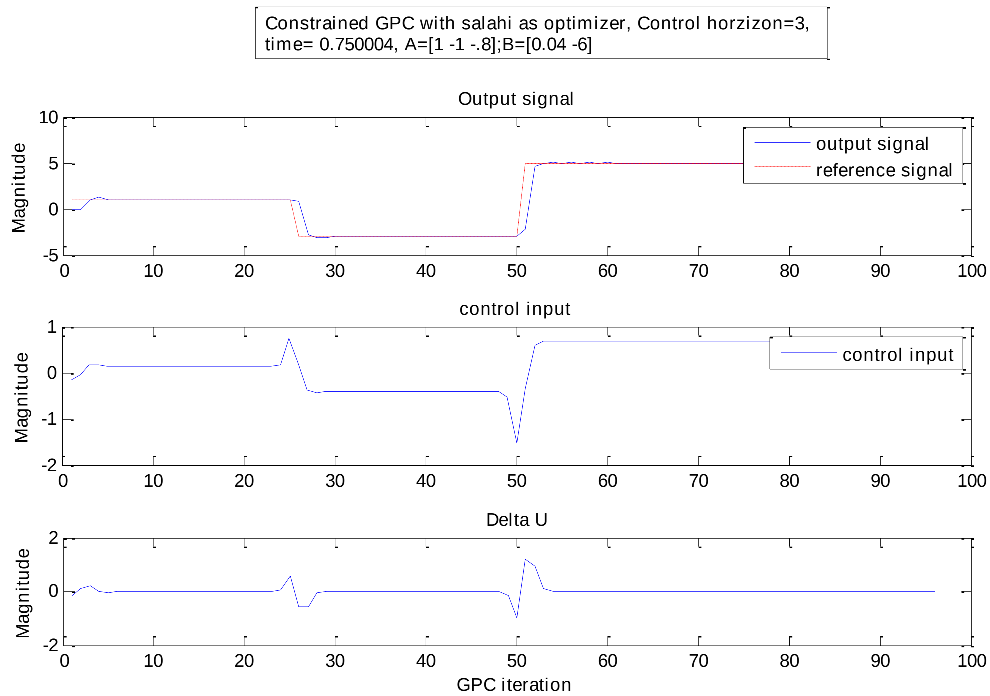

شکل ۴-۱۴: سیستم A= [1 -0.8]; B=[0.4 0.6]; با افق کنترل ۲۰ با بهینه ساز اصلاح شده‌ی مهروترا

### ۴-۲-۳-۲- سیستم ;A=[1 -1 -0.8]; B=[0.04 -6]

### افق کنترل برابر ۳:

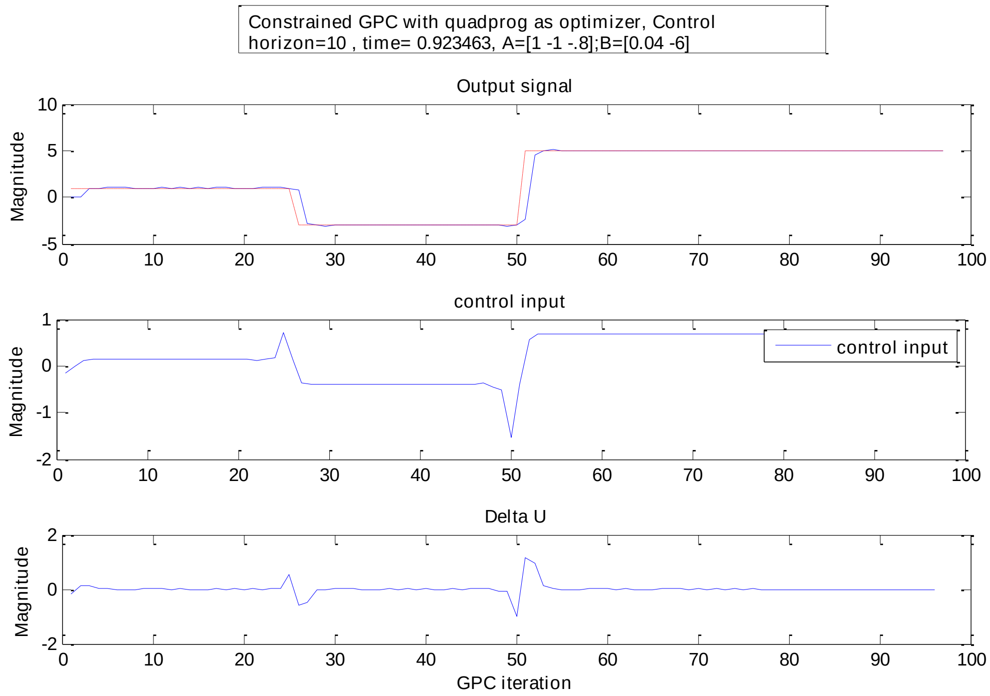



شکل ۴-۱۵: سیستم ;A= [1 -1 -0.8]; B=[0.04 -6] با افق کنترل ۳ با بهینه ساز متلب

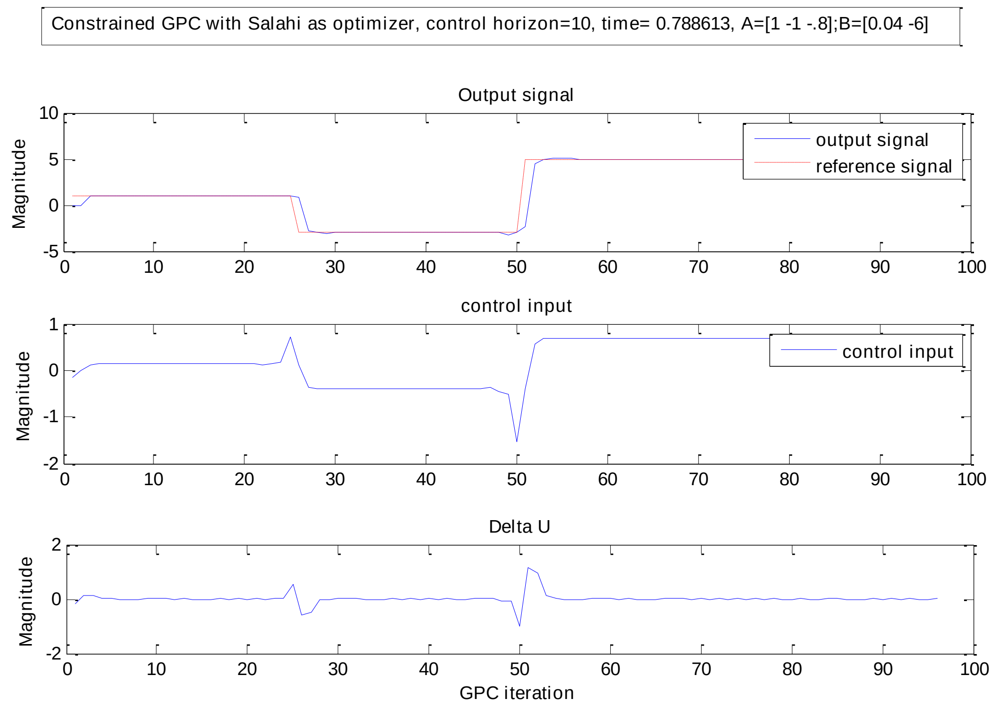



شکل ۴-۱۶: سیستم ;A= [1 -1 -0.8]; B=[0.04 -6] با افق کنترل ۳ با بهینه ساز اصلاح شده‌ی مهروترا

### افق کنترل برابر ۱۰:

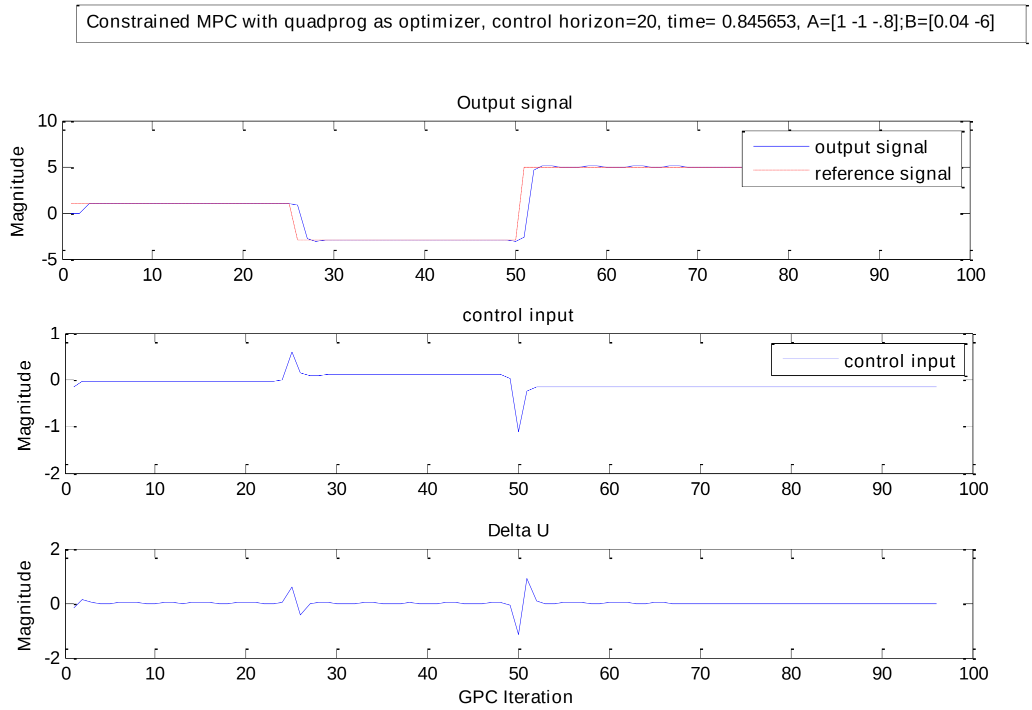



شکل۴-۱۷: سیستم ;[6- 0.04]=B ;[0.8- 1- 1] =A با افق کنترل ۱۰ با بهینه ساز متلب

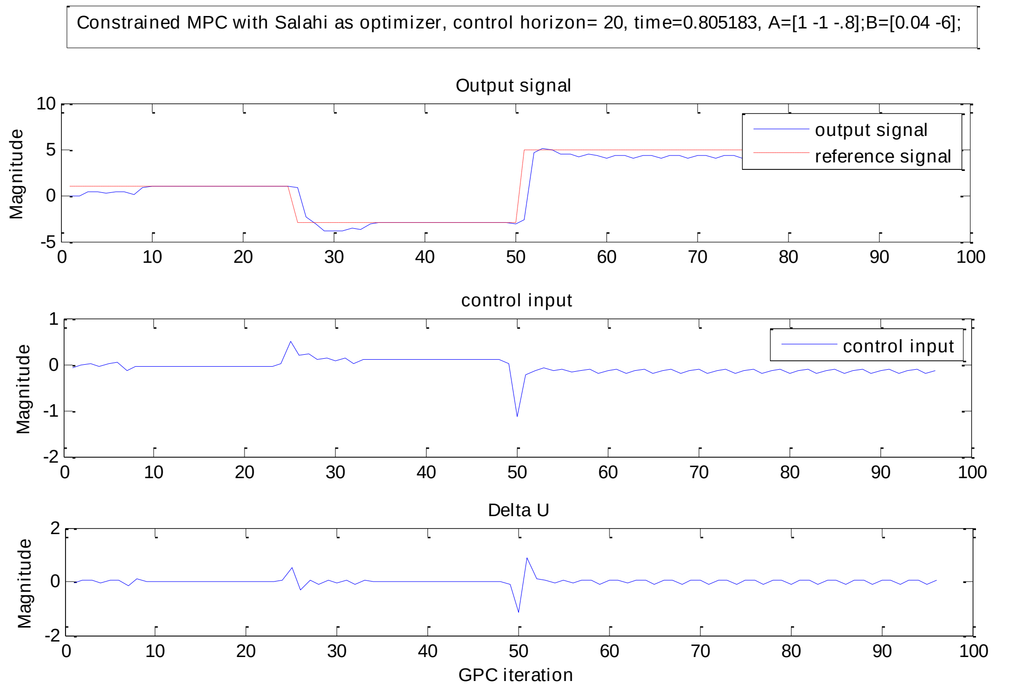



شکل۴-۱۸: سیستم ;[6- 0.04]=B ;[0.8- 1- 1] =A با افق کنترل ۱۰ با بهینه ساز اصلاح شده‌ی مهروترا

### افق کنترل برابر ۲۰:



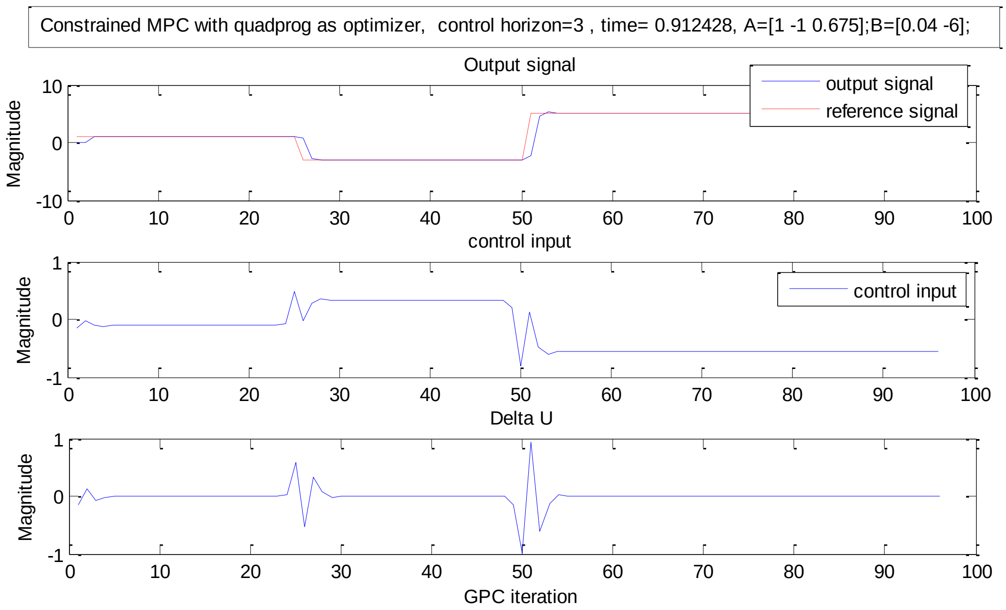

شکل ۴-۱۹: سیستم ;[6- 0.04]=B  ;[0.8- 1- 1] =A با افق کنترل ۲۰ با بهینه ساز متلب



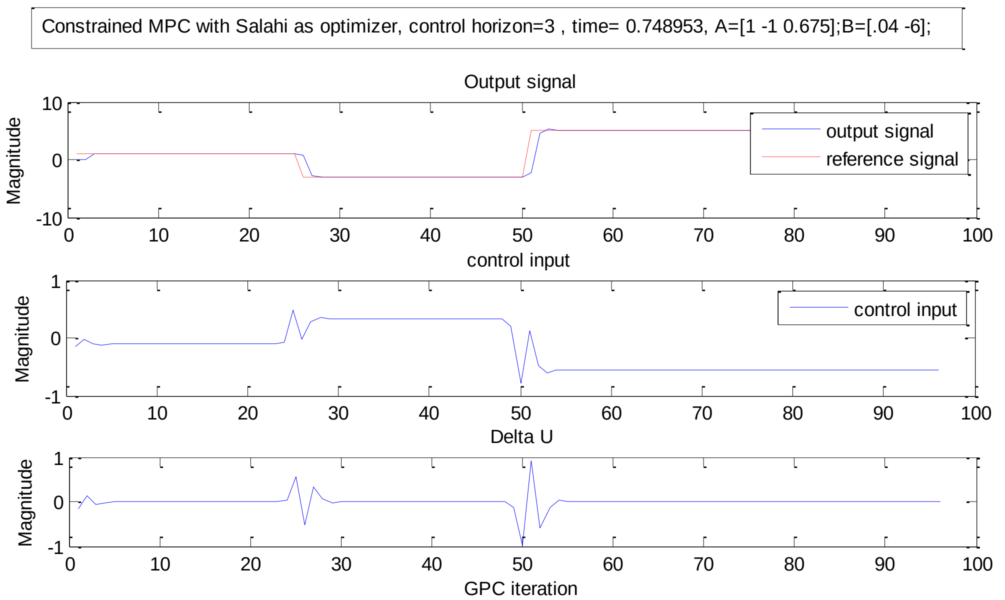

شکل ۴-۲۰: سیستم ;[6- 0.04]=B  ;[0.8- 1- 1] =A با افق کنترل ۲۰ با بهینه ساز اصلاح شده‌ی مهروترا

### ۴-۲-۳-۳- سیستم;[6- B=[0.04; [0.675 1- 1]=A

### افق کنترل برابر ۳:

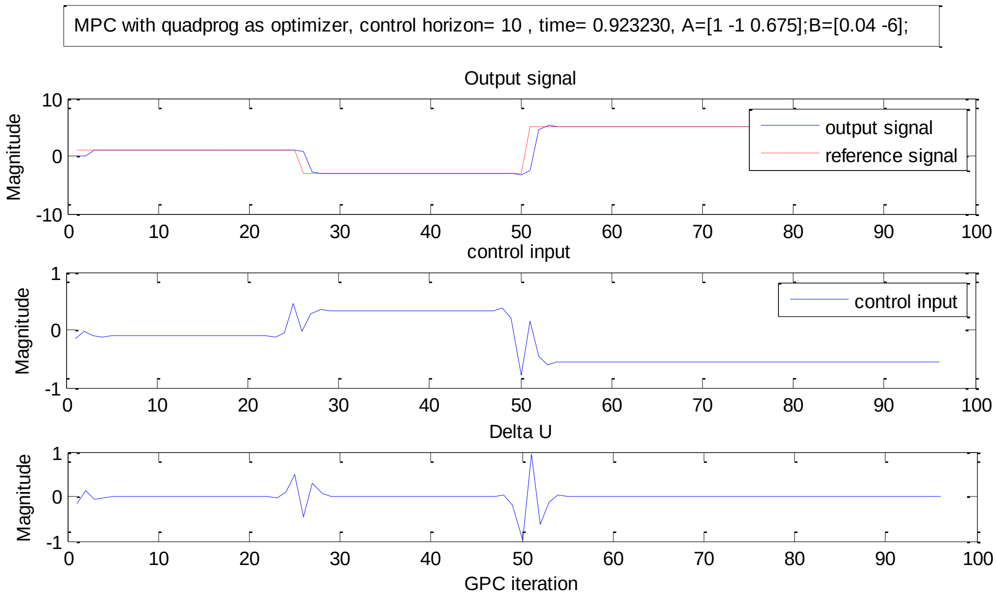



شکل ۴-۲۱: سیستم ;[6- B=[0.04 ;[0.675 1- 1] =A با افق کنترل ۳ با بهینه ساز متلب

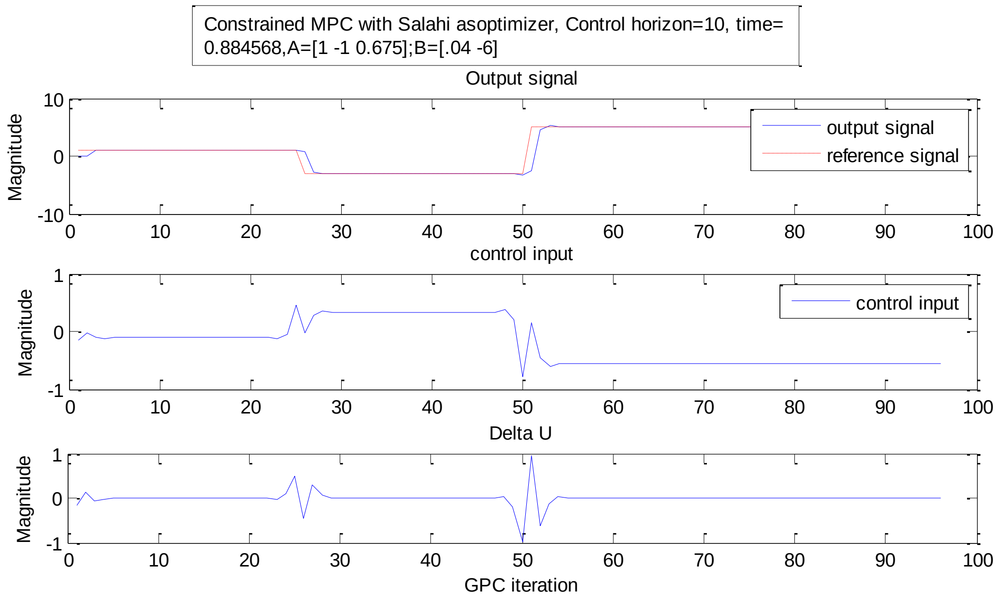



شکل ۴-۲۲: سیستم ;[6- B=[0.04 ;[0.675 1- 1] =A با افق کنترل ۳ با بهینه ساز اصلاح شده‌ی مهروترا

## افق کنترل برابر ۱۰:

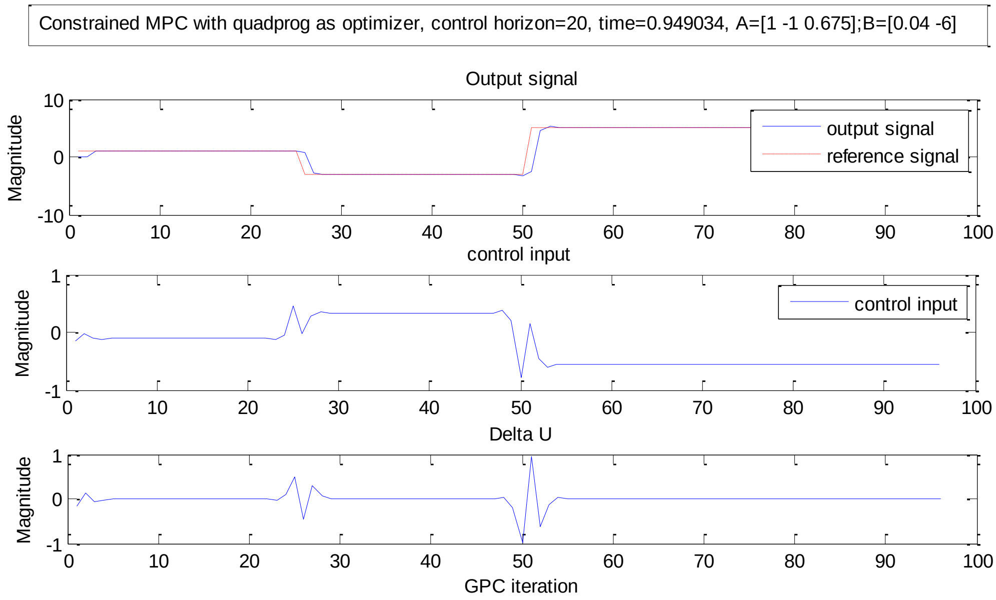



شکل ۴-۲۳: سیستم ;[6- B=[0.04  ;[0.675  1-  1] =A با افق کنترل ۱۰ با بهینه ساز متلب

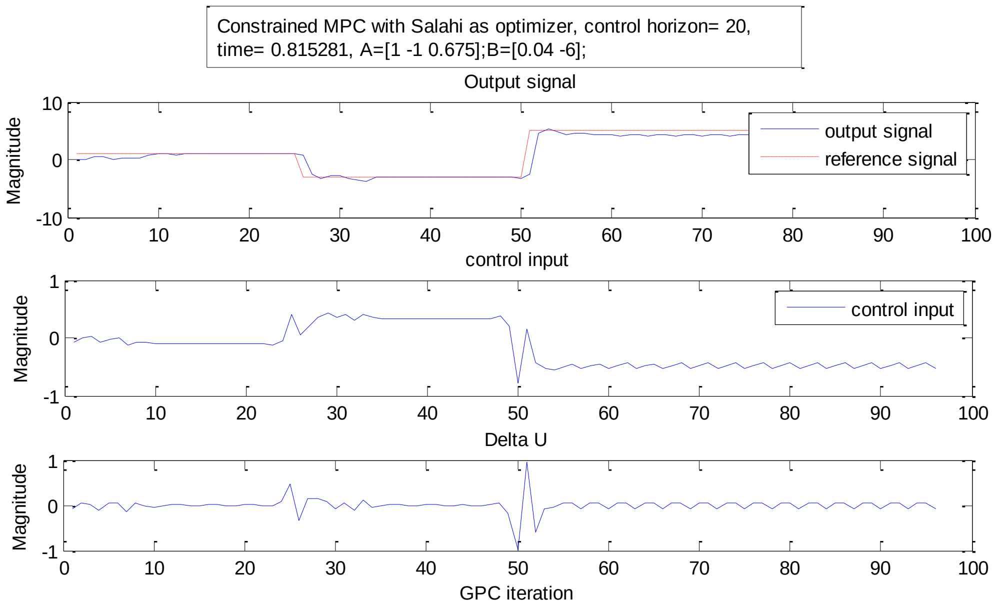



شکل ۴-۲۴: سیستم ;[6- B=[0.04  ;[0.675  1-  1] =A با افق کنترل ۱۰ با بهینه ساز اصلاح شده‌ی مهروترا

افق کنترل برابر ۲۰:



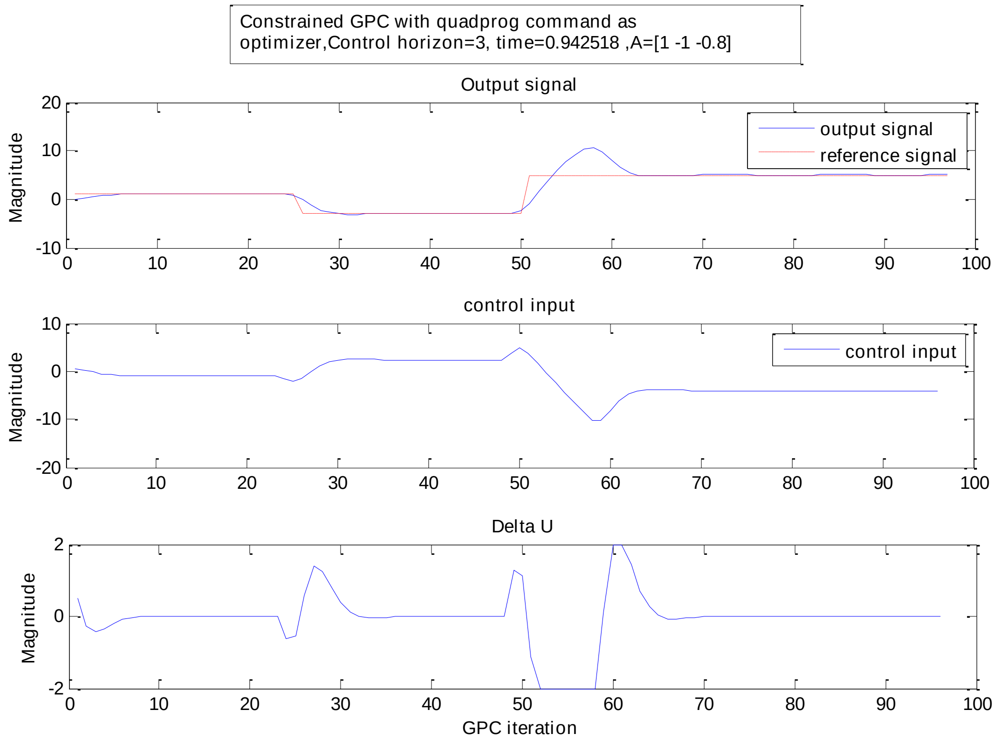

شکل ۴-۲۵: سیستم ;[6- B=[0.04  ;[0.675  1-  1] =A با افق کنترل ۲۰ با بهینه ساز متلب



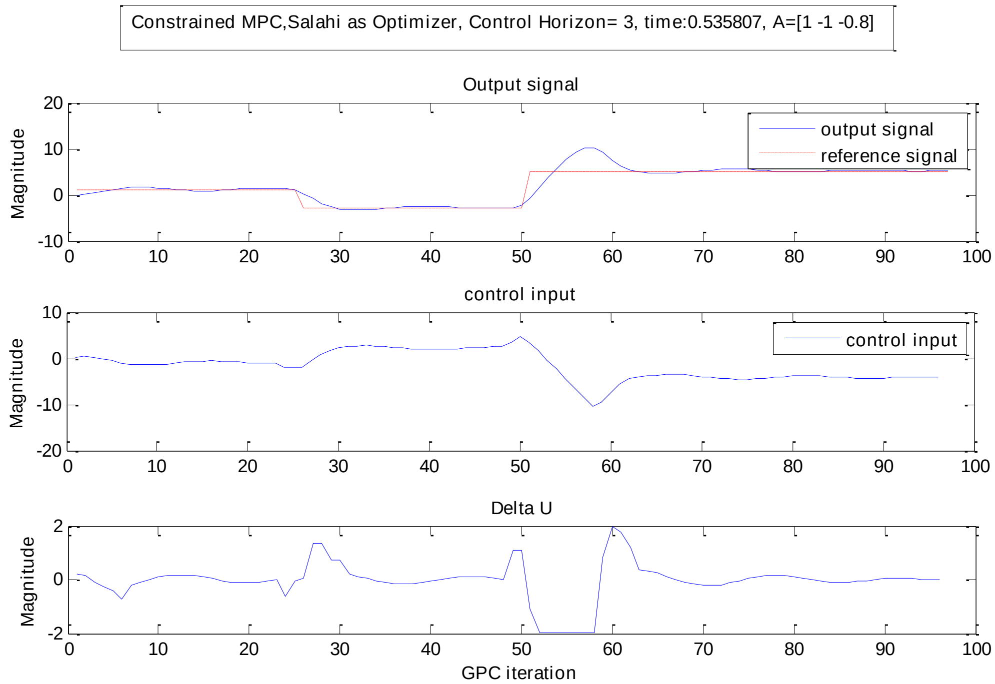

شکل ۴-۲۶: سیستم ;[6- B=[0.04  ;[0.675  1-  1] =A با افق کنترل ۲۰ با بهینه ساز اصلاح شده‌ی مهروترا

### ۴-۲-۳-۴-   سیستم A=[1  -1  -0.8]; B=[0.4  0.6];

### افق کنترل برابر ۳:

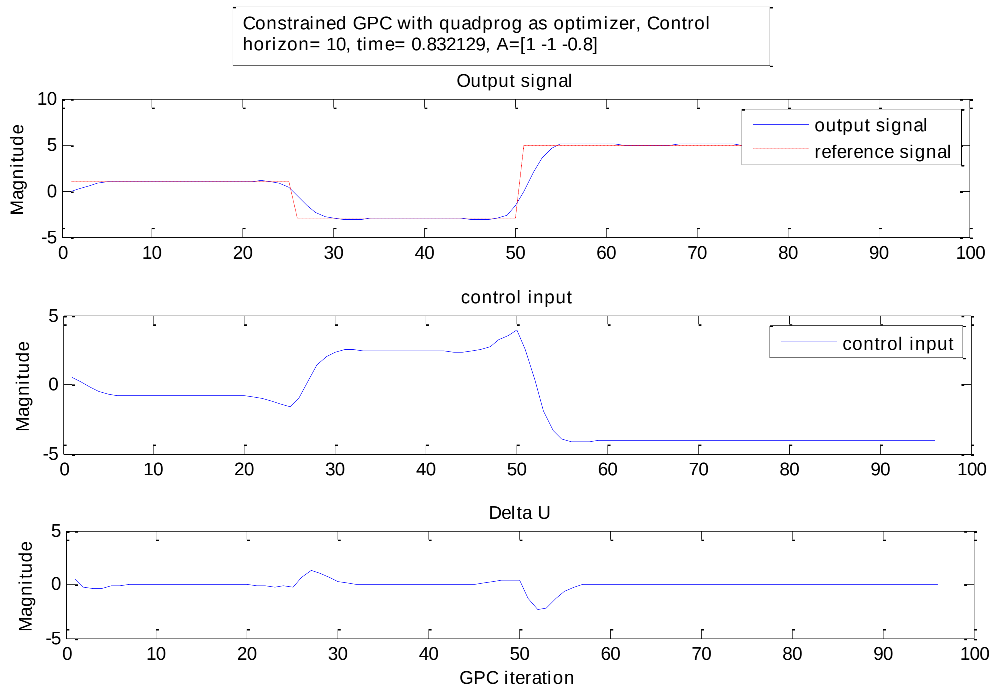



شکل۴-۲۷:سیستم A= [1  -1  -0.8];   B=[0.4  0.6]; با افق کنترل ۳ با بهینه ساز متلب

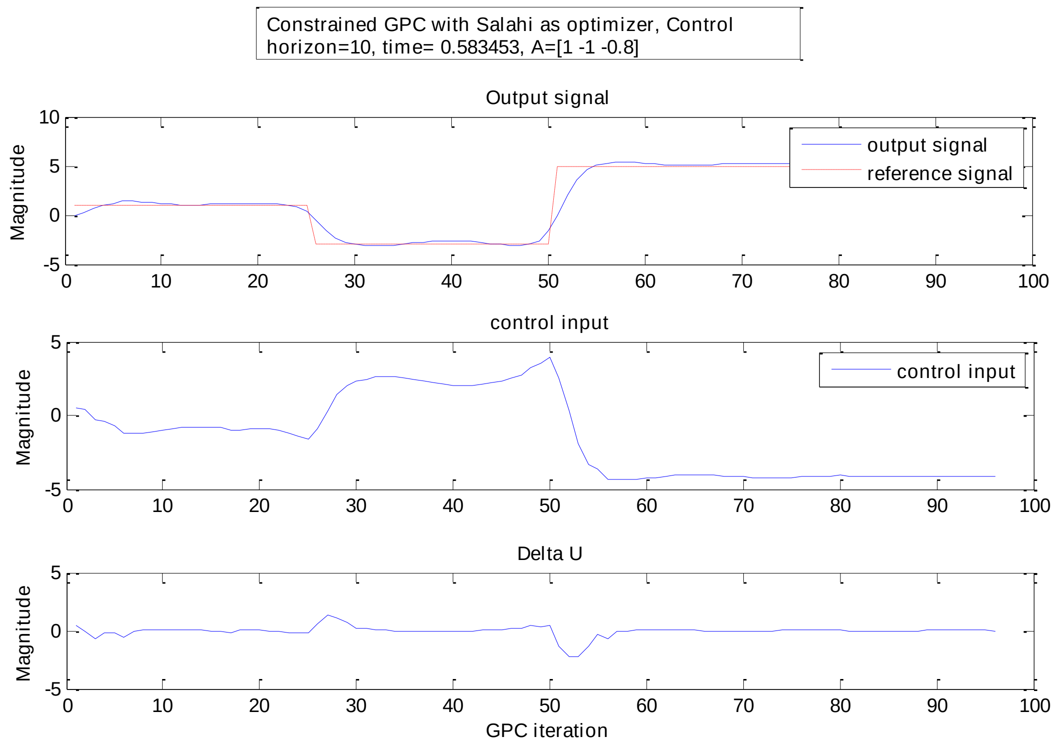



شکل۴-۲۸: سیستم A= [1  -1  -0.8];   B=[0.4  0.6]; با افق کنترل ۳ با بهینه ساز اصلاح شده‌ی مهروترا

## افق کنترل برابر ۱۰:

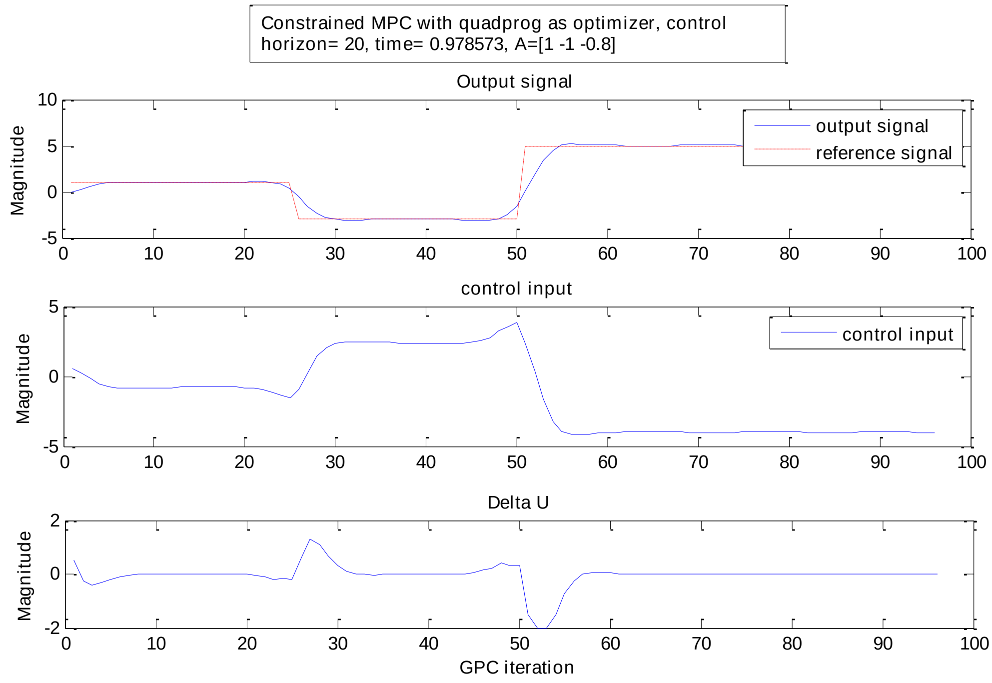



شکل۴-۲۹: سیستم A= [1 -1 -0.8]; B=[0.4 0.6]; با افق کنترل ۱۰ با بهینه ساز متلب

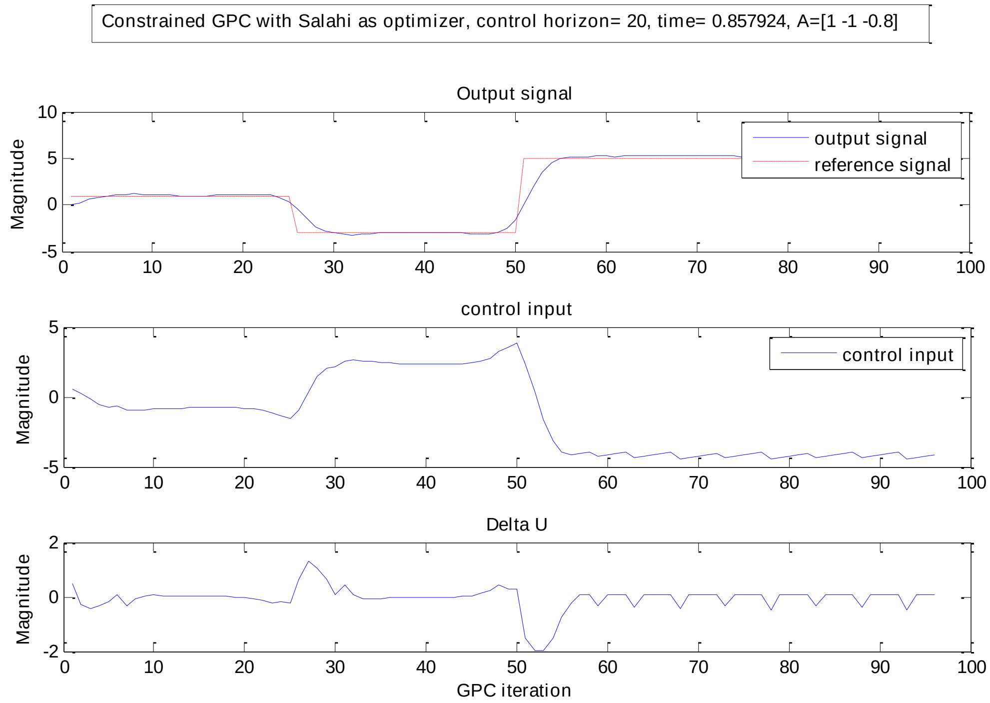



شکل۴-۳۰: سیستم A= [1 -1 -0.8]; B=[0.4 0.6]; با افق کنترل ۱۰ با بهینه ساز اصلاح شده‌ی مهروترا

افق کنترل برابر ۲۰:




شکل۴-۳۱: سیستم ;[0.6 0.4]=B ;[0.8- 1- 1] =A با افق کنترل ۲۰ با بهینه ساز متلب




شکل۴-۳۲: سیستم ;[0.6 0.4]=B ;[0.8- 1- 1] =A با افق کنترل ۲۰ با بهینه ساز اصلاح شده‌ی مهروترا

شکل‌های بالا از این جهت ارائه شده اند که نشان دهند زمان‌هایی که برای پاسخ‌گویی سیستم ذکر شد، با فرض کارایی قابل قبول سیستم بوده است. در غیر این‌صورت، بدیهی است که اگر عملکرد سیستم به ازای استفاده از الگوریتم اصلاح شده‌ی مهروترا کیفیت قابل قبولی نداشته باشد، سرعت زیاد محاسبات ارزشمند نخواهد بود.

# فصل ۵- نتیجه گیری و پیشنهادات

## ۵-۱- نتیجه گیری

یکی از مشکلات کنترل پیش بین، حجم زیاد محاسبات لازم برای حل مسئله‌ی بهینه سازی این روش است. با توجه به این مشکل، و این حقیقت که همین نقص، باعث محدود شدن کاربرد کنترل پیش بین شده است، پژوهش انجام شده در این پایان‌نامه، سعی در اعمال یک روش بهینه‌سازی سریع برای حل مسئله‌ی بهینه سازی کنترل پیش بین نموده است. پژوهش‌های انجام شده در این زمینه‌ی خاص بهینه سازی، منجر به انتخاب الگوریتم اصلاح شده‌ی مهروترا شد. پس از توسعه‌ی این روش برای مسئله‌ی برنامه ریزی درجه دوم، برنامه‌ی کامپیوتری برای شبیه سازی این الگوریتم نوشته شد و برای مثال‌های عددی گوناگون شبیه سازی صورت پذیرفت و سرعت پاسخ‌دهی الگوریتم با دستور بهینه‌سازی نرم افزار متلب مقایسه شد. با توجه به نتایج شبیه سازی‌های فصل چهارم، می‌توان گفت روش‌های نقطه‌ی درونی به طور کلی و گونه های مختلف الگوریتم پیش گو- اصلاح‌گر مهروترا به طور خاص، الگوریتم‌های مناسبی برای اعمال به مسئله‌ی کنترل پیش بین می‌باشند. به این معنا که با توجه به سرعت حل بالای این نوع الگوریتم‌ها نسبت به دیگر الگوریتم‌های موجود و با توجه به مشکلی که مسائل کنترل پیش بین با حجم محاسبات دارند، می‌توان در مسائل ابعاد وسیع از الگوریتم‌های نقطه‌ی درونی برای بهینه سازی استفاده کرد.

همچنین از آنجا که الگوریتم اصلاح شده‌ی مهروترا نیز دارای پاسخی سریع بود، می‌توان از این الگوریتم که معایب الگوریتم پیش‌گو-اصلاح‌گر مهروترا را برطرف کرده است در جهت بهبود سرعت عملکرد کنترل کننده های پیش بین در سیستم‌های با ابعاد وسیع استفاده کرد.

## ۵-۲- پیشنهادات

پیشنهادات متفاوتی را می‌توان برای ادامه‌ی این پایان نامه ارائه کرد. موارد زیر برای ادامه‌ی این پایان نامه پیشنهاد می‌گردد:

در توسعه‌ی الگوریتم اصلاح شده‌ی مهروترا (همچنین مهروترا و البته اکثر الگوریتم‌های بهینه سازی) شرط خروج از مسئله این است که بردار متغیرهای بهینه، که هدف یافتن مقدار کمینه یا بیشینه‌ی تابع معیار به ازای آن‌ها است، نسبت به تکرار قبلی الگوریتم تکرارشونده از مقدار معینی خطا کمتر تغییر کند. مثلاً در این پایان نامه برای خروج از الگوریتم بهینه سازی شرط زیر در نظر گرفته شده است :

$$norm\left(\begin{bmatrix} u(t+1|t) \\ u(t+2|t) \\ \vdots \\ u(t+N_c|t) \end{bmatrix}_k - \begin{bmatrix} u(t+1|t) \\ u(t+2|t) \\ \vdots \\ u(t+N_c|t) \end{bmatrix}_{k-1}\right) < \epsilon \qquad (۱-۵)$$

که در عبارت بالا اندیس $k$ نشان دهنده‌ی شماره‌ی تکرار الگوریتم بهینه سازی است (دقت کنید که با گام مسئله‌ی کنترل پیش بین نباید اشتباه شود که با $t$ نمایش داده شده است) و $N_c$ افق کنترلی است.

مقدار $\epsilon$ در اختیار طراح است. این پارامتر در طراحی و شبیه سازی بسیار اثرگذار است. نتیجه ای که بر اثر تکرارهای متوالی در این پایان نامه برای نویسنده حاصل شده است بدین قرار است که هر چه $\epsilon$ کوچک‌تر شود سرعت رسیدن به پاسخ برای مسئله‌ی بهینه سازی کمتر می‌شود زیرا مسئله‌ی بهینه سازی باید تکرارهای بیشتری را طی کند تا به پاسخ برسد و هرچه مقدار $\epsilon$ بزرگ‌تر شود اگرچه سرعت بیشتر می‌شود اما دقت مقدار سیگنال کنترلی محاسبه شده کاهش پیدا می‌کند. اگر این پارامتر از مقدار مشخصی بزرگ‌تر شود به طور کل مسئله‌ی کنترلی ناپایدار می‌شود (مثلاً برای مثال‌هایی که در این پایان نامه مطرح شده است این مقدار که منجر به ناپایداری می‌شود بزرگ‌تر از $10^{-2}$ است). بنابراین تعیین مقدار بهینه برای $\epsilon$ می‌تواند موضوع پژوهش‌های آینده باشد.

موضوع دیگری که می‌تواند مورد بررسی قرار گیرد مقدار مناسب قیود برای سیگنال کنترل است. با تکرارهای متوالی مشاهده شد که اگر مقدار سیگنال کنترلی از مقدار مشخصی کوچک‌تر در نظر گرفته شود، می‌تواند منجر به ناپایداری سیستم شود. این نتیجه‌ای است که منطقی به نظر می‌رسد، زیرا با محدود کردن سیستم به ورودی‌های کنترلی محدود عملاً آزادی عمل کنترل کننده در کنترل سیستم از بین رفته است. اما نکته‌ی تا اندازه‌ای قابل تأمل، این است که با افزایش نامحدود سیگنال کنترلی (که عملاً باید معادل نامقید کردن سیستم قلمداد شود) اگرچه سیستم پایدار است اما گاهی دچار نوسانات شدید و دائمی می‌گردد. این بدین معنی است که سیستم شاخص عملکرد ضعیفی دارد که در این موارد با محدودتر کردن مقدار سیگنال کنترل پاسخ بهبود می‌یابد. بر این اساس در پژوهش‌های آتی پیشنهاد می‌شود اثر محدود کردن سیگنال کنترلی بررسی گردد.

موضوعی دیگر، بررسی اعمال الگوریتم‌های بهینه سازی نامقید به سیستم و مقایسه‌ی پاسخ با پیدا کردن سیگنال کنترلی در حالت نامقید است. در مرجع [۴۳] مطرح شده است که با فرض نامقید بودن مسئله‌ی کنترل پیش بین پاسخ مسئله‌ی بهینه سازی زیر:

$$J = \frac{1}{2}u^T H u + B^T u + f_0$$

$$H = 2(G^T G + \lambda I)$$

$$b^T = 2(f-w)^T G$$

$$f_0 = (f-w)^T (f-w) \tag{۲-۵}$$

را می‌توان به سادگی با گرادیان گرفتن از $J$ به صورت زیر بدست آورد.

$$u = -H^{-1}b = (G^T G + \lambda I)^{-1} G^T (w-f) \tag{۳-۵}$$

ولی باید در نظر داشت که برای مسائلِ نامقید بهینه سازی هم الگوریتم‌های مخصوصی ارائه شده است. پیشنهاد می‌شود به مرجع [۲۳] برای اطلاعات بیشتر مراجعه شود. در میان روش‌های بهینه سازی نامقید، روش‌هایی مانند روش نیوتونی غیر دقیق[۱]، روش‌های شبه نیوتونی با حافظه محدود[۲] و به روزکردن شبه نیوتونی اسپارس[۳] به چشم می‌خورند که برای مسائل نامقید با ابعاد وسیع ارائه شده‌اند. در توضیح این روش‌ها ذکر شده است که مسائلی با هزاران یا میلیون‌ها متغیر را تنها زمانی می‌توان حل کرد که حافظه و هزینه‌ی محاسباتی الگوریتم بهینه سازی را بتوان در حد معقولی نگاه داشت. همچنین تابع معیار مسائلی این چنین بزرگ عموماً دارای ساختار به صورتی خاص هستند که به عنوان "قابلیت جداسازی تکه ای"[۴] شناخته می‌شود که به این معنا است که قابلیت این را دارند که به صورت جمع توابع ساده تر نوشته شوند، هر کدام از این توابع به قسمت کوچکی از یک زیر فضای $\mathbb{R}^n$ تعلق دارند. بنابراین پیشنهاد می‌شود در این‌گونه مسائل با ابعاد بسیار بزرگ این الگوریتم‌های بهینه سازی نامقید با حل تحلیلی به صورت فرمولی که در بالا ذکر شد مقایسه گردد. از آنجا که مسئله‌ی کنترل پیش بین یک مسئله‌ی صنعتی است ممکن است دارای ابعاد بسیار وسیع باشد که این مقایسه –اگرچه مسئله را نامقید فرض کرده است- می‌تواند نتایج درخور توجهی داشته باشد.

[1]Inexact Newton Methods
[2]Limited-Memory Quasi-Newton methods
[3]Sparse Quasi-Newton Update
[4]Partial separability

# فهرست مراجع


[1] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica,* vol. 36, pp. 789-814, 2000.

[2] C. V. Rao and J. B. Rawlings, "Linear programming and model predictive control," *Journal of Process Control,* vol. 10, pp. 283-289, 2000.

[3] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice,* vol. 11, pp. 733-764, 2003.

[4] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of Interior-Point Methods to Model Predictive Control," *Journal of Optimization Theory and Applications,* vol. 99, pp. 723-757, 1998.

[5] D. Chmielewski and V. Manousiouthakis, "On constrained infinite-time linear quadratic optimal control," *Systems & Control Letters,* vol. 29, pp. 121-129, 1996.

[6] P. O. M. Scokaert and J. B. Rawlings, "Constrained linear quadratic regulation," *Automatic Control, IEEE Transactions on,* vol. 43, pp. 1163-1169, 1998.

[7] M. Sznaier and M. J. Damborg, "Suboptimal control of linear systems with state and control inequality constraints," in *Decision and Control, 1987. 26th IEEE Conference on*, 1987, pp. 761-762.

[8] J. L. Jerez, E. C. Kerrigan, and G. A. Constantinides, "A sparse and condensed QP formulation for predictive control of LTI systems," *Automatica,* vol. 48, pp. 999-1002, 2012.

[9] A. Domahidi, A. Zgraggen, M. N. Zeilinger, M. Morari, and C. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *Proc. of the 51st Conference on Decision and Control, Maui, Hawaii, USA*, 2012.

[10] P. Škrabánek and D. Honc, "Fast Optimization Algorithm for Constrained Model Predictive Control," 2003.

[11] F. Borrelli, M. Baotić, J. Pekar, and G. Stewart, "On the computation of linear model predictive control laws," *Automatica,* vol. 46, pp. 1035-1041, 2010.

[12] C. Onnen, R. Babuika, U. Kaymak, J. Sousa, H. Verbruggen, and R. Isermann, "Genetic algorithms for optimization in predictive control," *Control Engineering Practice,* vol. 5, pp. 1363-1372, 1997.

[13] A. Draeger, S. Engell, and H. Ranke, "Model predictive control using neural networks," *Control Systems, IEEE,* vol. 15, pp. 61-66, 1995.

[14] R. Bartlett, A. Wachter, and L. Biegler, "Active set vs. interior point strategies for model predictive control," in *American Control Conference, 2000. Proceedings of the 2000*, 2000, pp. 4229-4233.

[15] S. J. Wright, "Applying new optimization algorithms to model predictive control," in *AIChE Symposium Series*, 1997, pp. 147-155.

[16] M. Kogel and R. Findeisen, "Fast predictive control of linear, time-invariant systems using an algorithm based on the fast gradient method and augmented Lagrange multipliers," in *Control Applications (CCA), 2011 IEEE International Conference on*, 2011, pp. 780-785.

[17] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *Control Systems Technology, IEEE Transactions on,* vol. 18, pp. 267-278, 2010.

[18] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica,* vol. 38, pp. 3-20, 2002.

[19] A. Shahzad, E. C. Kerrigan, and G. A. Constantinides, "A warm-start interior-point method for predictive control," *Proc. UKACC,* 2010.

[20] A. Shahzad, E. C. Kerrigan, and G. A. Constantinides, "A fast well-conditioned interior point method for predictive control," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, 2010, pp. 508-513.

[21] A. Shahzad and P. J. Goulart, "A New Hot-start Interior-point Method for Model Predictive Control," in *World Congress*, 2011, pp. 2470-2475.

[22] E. A. Yildirim and S. J. Wright, "Warm-start strategies in interior-point methods for linear programming," *SIAM Journal on Optimization,* vol. 12, pp. 782-810, 2002.

[23] J. Nocedal and S. J. Wright, "Numerical Optimization," *Springer Series in Operations Research and Financial Engineering (*2006).

[24] R. J. Vanderbei, "LOQO: An interior point code for quadratic programming," *Optimization methods and software,* vol. 11, pp. 451-484, 1999.

[25] F. Delbos and J. C. Gilbert, "Global Linear Convergence of an Augmented Lagrangian Algorithm to Solve Convex Quadratic Optimization Problems," *Journal of Convex Analysis,* vol. 12, pp. 045-069, 2005.

[26] P. Wolfe, "The simplex method for quadratic programming," *Econometrica: Journal of the Econometric Society,* pp. 382-398, 1959.

[27] G. Dantzig, *Linear programming and extensions*: Princeton university press, 1998.

[28] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica,* vol. 4, pp. 373-395, 1984.

[29] V. Klee and G. J. Minty, "How Good Is the Simplex Algorithm?," *Inequalities: proceedings,* vol. 3, pp. 159-175, 1972.

[30] S. Boyd and L. Vandenberghe, *Convex optimization*: Cambridge university press, 2004.

[31] Y. Koohmaskan, “Convex Optimization in Model Predictive Control, Final Project of Convex Optimization,” On Internet: http://koohmaskan.persiangig.com/document/Homepage/mpc_cvx.pdf, August 25, 2010.

[32] S. J. Wright, *Primal-Dual Interior-Point Methods*: Society for Industrial and Applied Mathematics, 1997.

[33] S. Mehrotra, "On the implementation of a primal-dual interior point method," *SIAM Journal on Optimization,* vol. 2, pp. 575-601, 1992.

[34] E. D. Andersen and K. D. Andersen. "The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm." *High performance optimization*, vol. 33, pp. 197-232, 2000.

[35] CPLEX: ILOG Optimization. On Internet: http://www.ilog.com.

[36] J. Czyzyk, S. Mehrotra, M. Wagner, and S. J. Wright, "PCx: An interior-point code for linear programming," *Optimization methods and software,* vol. 11, pp. 397-430, 1999.

[37] XpressMP: Dash Optimization. On Internet: http://www.dashoptimization.com.

[38] Y. Zhang, "Solving large-scale linear programs by interior-point methods under the Matlab—Environment, " *Optimization methods and software,* vol. 10, pp. 1-31, 1998.

[39] X. Zhu, J. Peng, T. Terlaky, and G. Zhang. “On implementing self-regular proximitybased feasible IPMs.” Tech. Rep. 2003/2, Advanced Optimization Lab. Department of Computing and Software, McMaster University, Hamilton, ON, Canada. On Internet: http://www.cas.mcmaster.ca/~oplab/publication.

[40] M. Salahi, J. Peng, and T. Terlaky, "On Mehrotra-Type Predictor-Corrector Algorithms," *SIAM Journal on Optimization,* vol. 18, pp. 1377-1397, 2008.

[41] M. Salahi and T. Terlaky, "Mehrotra-type predictor-corrector algorithm revisited," *Optimization methods and software,* vol. 23, pp. 259-273, 2008.

[42] Aspentech DMCplus family brochure, code: 11-391-0411, On Internet:
http://www.aspentech.com/WorkArea/DownloadAsset.aspx?id=6442451701

[43] E. Camacho and C. Bordons, *Model Predictive Control*: Springer Verlag, 2004.

[44] D. W. Clarke, C. Mohtadi, and P. Tuffs, "Generalized predictive controlâ€”Part I. The basic algorithm," *Automatica,* vol. 23, pp. 137-148, 1987.

[45] M. Salahi, "New Adaptive Interior Point Algorithms for Linear Optimization, " Ph.D. Dissertation, Dept. of Math., McMaster University, Canada, March 2006.

[46] B. LieW, M. D. Diez, and T. A. Hauge, "A comparison of implementation strategies for MPC," *Roll-out of model based control with application to paper machines,* p. 192, 2005.

[47] K. R. Muske and J. B. Rawlings, "Model predictive control with linear models," *AIChE Journal,* vol. 39, pp. 262-287, 1993.

[48] J. B. Rawlings and K. R. Muske, "The stability of constrained receding horizon control," *Automatic Control, IEEE Transactions on,* vol. 38, pp. 1512-1516, 1993.

[49] J. R. Gilbert, C. Moler, and R. Schreiber, "Sparse matrices in MATLAB: design and implementation," *SIAM Journal on Matrix Analysis and Applications,* vol. 13, pp. 333-356, 1992.

[50] T. F. Coleman and Y. Li, "A reflective Newton method for minimizing a quadratic function subject to bounds on some of the variables," *SIAM Journal on Optimization,* vol. 6, pp. 1040-1058, 1996.

[51] P. E. Gill, W. Murray, and M. H. Wright, *Practical optimization*: Academic Pr, 1981.

[52] MATLAB Optimization Toolbox Version 5.0 Natick, Massachusetts: The MathWorks Inc., 2010.

# واژه نامه فارسی به انگلیسی

| | |
|---|---|
| ابتکار | Heuristic |
| ابعاد وسیع | Large-scale |
| اسپارس، تُنُک | Sparse |
| افق پیش بینی | Prediction Horizon |
| الگوریتم پیش گو-اصلاح گر اولیه-دوگان نقطه درونی | Predictor-corrector primal-dual interior-point method |
| الگوریتم پیش گو اصلاح گر | Predictor-Corrector Algorithm |
| الگوریتم دانتزیگ-ولف | Dantzig-Wolfe's algorithm |
| انحراف کنترل | Control deviation |
| اهداف بولزا | Bolza Objective |
| اولیه دوگان | Primal-dual |
| ایستا | Stationary |
| اینونسیس | Invensys |
| بردار کمکی | Slack vector |
| برنامه ریزی درجه دوم | Quadratic programming |
| بهینه سازی موضعی و سراسری | Local & Global Optimization |
| تصویر گرادیان | Gradient projection |
| تغییرات سیگنال کنترلی | Control increment |
| تلاش کنترلی | Control Effort |
| جدول مرجع | Lookup table |
| جعبه ابزار کنترل پیش بین | Model predictive control toolbox |
| جملهی مرکزی | Centering Parameter |
| جهت مقیاس بندیِ آفینی پیش گو | Affine-Scaling Predictor Direction |
| جهت مقیاس بندی آفینی | Affine scaling direction |
| چگال | Dense |
| رگولاتور درجه دوم خطی | Linear Quadratic Regulator |
| بروزکردن شبه نیوتونی اسپارس | Sparse Quasi-Newton Update |
| روش بیضی خاچیان | Khachian's ellipsoid method |
| روش تصویری | Projection method |
| روشهای با گام بلند | Long-step methods |
| روشهای پیش گو اصلاح گر | Predictor-Corrector Methods |
| روشهای حامل | Barrier methods |
| روشهای مسیریاب | Path-Following methods |
| روشهای مسیریاب اولیه-دوگان | Primal-Dual Path following algorithms |
| روشهای شبه نیوتونی با حافظه محدود | Limited-Memory Quasi-Newton methods |

| | |
|---|---|
| Short-step methods | روش‌های گام کوتاه |
| Interior-point methods | روش‌های نقطه درونی |
| Dead time | زمان مرده |
| Sub-optimal | زیر بهینه |
| Nominally stable closed loop system | سیستمِ حلقه بسته‌ی پایدار اسمی |
| Simplex | سیمپلکس |
| Feasible | شدنی |
| Strictly feasible | شدنی مطلق |
| Linear Independence Constraint Qualification | شرایط قیدهای خطی مستقل |
| Warm start | شروع گرم |
| Salahi-Terlaki | صلاحی- ترلکی |
| Actuators | عملگرها |
| QR factorization | تجزیه کردن به صورت QR |
| Partial separability | قابلیت جداسازی تکه ای |
| Hard Constraints | قیدهای سخت |
| Soft Constraints | قیدهای نرم |
| Performance | کارایی |
| Karmarker | کارمارکر |
| Karush-Kuhn-Tucker | کاروش کیون تاکر |
| Weak Local Minimizer | کمینه‌ی موضعی ضعیف |
| Generalized Predictive Control | کنترل پیش بین تعمیم یافته |
| Explicit Model predictive control | کنترل پیش بین صریح |
| Convexity | محدب بودن |
| Gradient | گرادیان |
| Conjugate gradient | گرادیان الحاقی |
| Augmented Lagrangian | لاگرانژین افزوده |
| Subspace trust-region method | متد زیر فضای قابل اعتماد |
| Interior-reflective Newton method | متد نیوتون انعکاس دهنده‌ی میانی |
| Active-set methods | متدهای مجموعه فعال |
| Orthogonal | متعامد |
| Inexact Newton Methods | متد نیوتونی غیر دقیق |
| Feasible Set | مجموعه شدنی |
| Strictly feasible set | مجموعه شدنی مطلق |
| Active Set | مجموعه فعال |
| Critical Cone | مخروط بحرانی |
| Controller Auto-Regressive Moving-Average | مدل کنترل کننده با میانگین متحرک رگرسیو خودکار |
| Desired Reference | مرجع مطلوب |
| Centrality | مرکزیت |
| Convex Quadratic Program | مسئله درجه دوم محدب |
| Strictly convex QP | مسائل برنامه ریزی درجه دوم محدبِ اکید |
| Reference Trajectory | مسیر مرجع |

| | |
|---|---|
| مسیر مرکزی | Central Path |
| معادله دیوفانتین | Diophantine equation |
| معیار دوگانی | Duality Measure |
| مکملی | Complementary |
| مهروترا | Mehrotra |
| ناشدنی | Infeasible |
| نرم افزار LIPSOL | Linear Interior-Point solver |

| | |
|---|---|
| نقطه شدنی | Feasible point |
| نقطه‌ی کمینه‌ی موضعی تنها | Isolated Local Minimizer |
| هادامارد | Hadamard |
| هسین | Hessian |
| هموار | Smooth |

# واژه نامه انگلیسی به فارسی

| | |
|---|---|
| Active Set | مجموعه فعال |
| Active-set methods | متدهای مجموعه فعال |
| Actuators | عملگرها |
| Affine scaling direction | جهت مقیاس بندی آفینی |
| Affine-Scaling Predictor Direction | جهت مقیاس بندیِ آفینی پیش گو |
| Augmented Lagrangian | لاگرانژین افزوده |
| Barrier methods | لاگرانژین افزوده |
| Bolza Objective | اهداف بولزا |
| Centering Parameter | جملهی مرکزیت |
| Central Path | مسیر مرکزی |
| Centrality | مرکزیت |
| Complementary | مکملی |
| Conjugate gradient | گرادیان الحاقی |
| Control deviation | انحراف کنترل |
| Control Effort | تلاش کنترلی |
| Control increment | تغییرات سیگنال کنترلی |
| Controller Auto-Regressive Moving-Average | مدل کنترل کننده با میانگین متحرک رگرسیو خودکار |
| Convex Quadratic Program | مسئله درجه دوم محدب |
| Convexity | محدب بودن |
| Critical Cone | مخروط بحرانی |
| Dantzig-Wolfe’s algorithm | الگوریتم دانتزیگ-ولف |
| Dead time | زمان مرده |
| Dense | چگال |
| Desired Reference | مرجع مطلوب |
| Diophantine equation | معادله دیوفانتین |
| Duality Measure | معیار دوگانی |
| Explicit Model predictive control | کنترل پیش بین صریح |
| Feasible | شدنی |
| Feasible point | نقطه شدنی |
| Feasible Set | مجموعه شدنی |
| Generalized Predictive Control | کنترل پیش بین تعمیم یافته |
| Gradient | گرادیان |
| Gradient projection | تصویر گرادیان |
| Hadamard | هادامارد |
| Hard Constraints | قیدهای سخت |
| Hessian | هسین |
| Heuristic | ابتکار |
| Inexact Newton Methods | روش نیوتونی غیر دقیق |
| Infeasible | ناشدنی |
| Interior-point methods | روشهای نقطه درونی |

| | |
|---|---|
| Interior-reflective Newton method | روش نیوتون انعکاس دهنده‌ی میانی |
| Invensys | اینونسیس |
| Isolated Local Minimizer | نقطه‌ی کمینه‌ی موضعی تنها |
| Karmarker | کارمارکر |
| Karush-Kuhn-Tucker | کاروش کیون تاکر |
| Khachian’s ellipsoid method | روش بیضی خاچیان |
| Large-Scale | ابعاد وسیع |
| Limited-Memory Quasi-Newton methods | روش‌های شبه نیوتونی با حافظه محدود |
| Linear Independence Constraint Qualification | شرایط قیدهای خطی مستقل |
| Linear Interior-Point solver | نرم افزار LIPSOL |
| Linear Quadratic Regulator | رگولاتور درجه دوم خطی |
| Local & Global Optimization | بهینه سازی موضعی و سراسری |
| Long-step methods | روش‌های با گام بلند |
| Lookup table | جدول مرجع |
| Mehrotra | مهروترا |
| Model predictive control toolbox | جعبه ابزار کنترل پیش بین |
| Nominally stable closed loop system | سیستمِ حلقه بسته‌ی پایدار اسمی |
| Orthogonal | متعامد |
| Partial separability | قابلیت جداسازی تکه ای |
| Path-Following methods | روش‌های مسیریاب |
| Performance | کارایی |
| Prediction Horizon | افق پیش بینی |
| Predictor-Corrector Algorithm | الگوریتم پیش گو اصلاح گر |
| Predictor-Corrector Methods | روش‌های پیش گو اصلاح گر |
| Predictor-corrector primal-dual interior-point method | الگوریتم پیش گو-اصلاح گر اولیه-دوگان نقطه درونی |
| Primal-dual | اولیه دوگان |
| Primal-Dual Path following algorithms | روش‌های مسیریاب اولیه-دوگان |
| Projection method | روش تصویری |
| QR factorization | تجزیه کردن به صورت QR |
| Quadratic programming | برنامه ریزی درجه دوم |
| Reference Trajectory | مسیر مرجع |
| Salahi-Terlaki | صلاحی- ترلکی |
| Short-step methods | روش‌های گام کوتاه |
| Simplex | سیمپلکس |
| Slack vector | بردار کمکی |
| Smooth | هموار |
| Soft Constraints | قیدهای نرم |

| | |
|---|---|
| Sparse | اسپارس، تُنُک |
| Sparse Quasi-Newton Update | بروزکردن شبه نیوتونی اسپارس |
| Stationary | ایستا |
| Strictly convex QP | مسائل برنامه ریزی درجه دوم محدبِ اکید |
| Strictly feasible | شدنی مطلق |
| Strictly feasible set | مجموعه شدنی مطلق |
| Sub-optimal | زیر بهینه |
| Subspace trust-region method | روش زیر فضای قابل اعتماد |
| Warm start | شروع گرم |
| Weak Local Minimizer | مینیمم کننده‌ی موضعی ضعیف |

**Abstract**

In this thesis, we focused on the optimization problem in model predictive control. In model predictive control, a convex quadratic program should be solved to obtain control input, and since the problem is constrained, constrained convex quadratic programming algorithms are eligible to be implemented. There are different methods for solving optimization problem of model predictive control, in this thesis, we focused on a unique algorithm, named Mehrotra, and after discussing advanteges and properties, a variant of this algorithm first were developed for convex quadratic programming problem and then is is applied to model predictive control problem, at last with presenting different examples as model predictive control system, this algorithm is applied to them with different horizons, and after that the result is compared with MATLAB's special quadratic programming command(optimization toolbox version 5.0). The results show that implementing the variant of Mehrotra's algorithm will result faster responses and computations.

K. N. Toosi University of Technology
Facultyof Electrical and Computer Engineering
Department of Systems and Control

# Design of Fast Model Predictive Control Systems Using Advanced Optimization Algorithms

ThesisSubmitted in Partial Fulfillment of the
Requirements for the Degree of Master of Science (M.Sc.)
in Electrical Engineering, Systems and Control.

By:
**Saman Cyrus**

Supervisor:
**Professor A. Khaki-Sedigh**

Advisor:
**Dr. M. R. Peyghami**

Winter 2013