ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Σχολή: Χημικών Μηχανικών

Τομέας Ανάλυσης, Σχεδιασμού και Ανάπτυξης Διεργασιών και Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μελέτη Αλγόριθμων Τετραγωνικού Προγραμματισμού για Γρήγορη Επίλυση Προβλημάτων Προβλεπτικού Ελέγχου

Όνομα: Παπαθανασόπουλος Σωτήρης

Επιβλέπων Καθηγητής: Σαρίμβεης Χαράλαμπος

Αθήνα, Ιούλιος 2013

Περίληψη:

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη κώδικα για την υλοποίηση της μεθόδου Επιταχυνόμενης Δυικής Προβολής Κλίσης (GPAD) με σκοπό την γρήγορη επίλυση προβλημάτων τετραγωνικού προγραμματισμού που προκύπτουν κατά την διαμόρφωση προβλημάτων Προβλεπτικού ελέγχου (MPC) για την βέλτιστη ρύθμιση συστημάτων. Επίσης δημιουργούμε και τον κώδικα ο οποίος διαμορφώνει το πρόγραμμα τετραγωνικού προγραμματισμού .
Αναλύουμε τις συνθήκες στις οποίες στηρίζεται η θεωρία κυρτής βελτιστοποίησης, μελετούμε τον τρόπο υλοποίησης διαφόρων αλγορίθμων επίλυσης τέτοιων προβλημάτων και συγκρίνουμε τις αποδόσεις μερικών από αυτών.
Ο κώδικας αναπτύχθηκε στο λογισμικό Matlab και η πειραματική μελέτη αφορούσε την σύγκριση, χρόνου και αποτελεσματικότητας επίλυσης των προβλημάτων από τους αλγόριθμους: QUADPROG, GPAD και CPLEX. Οι συγκρίσεις έγιναν σε benchmark προβλήματα προβλεπτικού ελέγχου που υπάρχουν στην ιστοσελίδα:
http://www.kuleuven.be/optec/software/onlineQP?start=1
καθώς και σε ένα πρόβλημα δικής μας κατασκευής.
Στο τέλος αναφέρουμε τα συμπεράσματα των αποτελεσμάτων μας καθώς και τους τομείς οπού θα έπρεπε να υπάρξει περαιτέρω έρευνα για την αποτελεσματικότερη αντιμετώπιση προβλημάτων κατά την εφαρμογή MPC.

Περιεχόμενα:

1 Εισαγωγή:

Ο προβλεπτικός έλεγχος (Model. Predictive. Control) είναι μια γνωστή μεθοδολογία για τη σύνθεση νόμων ελέγχου, που βελτιστοποιούν την απόδοση κλειστού βρόγχου η οποία υπόκεινται σε προκαθορισμένους λειτουργικούς περιορισμούς στις εισόδους, στις μεταβλητές κατάστασης και στις εξόδους. Κάποιες φορές τον συναντούμε και με άλλη ονομασία πέραν του MPC όπως :

- Δυναμικός έλεγχος πινάκων (Dynamic matrix control)
- Εναλλασσόμενος έλεγχος ορίζοντα (receding horizon control)
- Δυναμικός Γραμμικός Προγραμματισμός
- Σχεδιασμός μεταβλητού ορίζοντα (rolling horizon planning)

Αυτές οι δυνατότητες προσέλκυσαν ιδιαίτερα τα τελευταία χρόνια την αεροδιαστημική βιομηχανία για μία ποικιλία διαφορετικών εφαρμογών. Κυρίως πάντως μέχρι σήμερα συναντάται σε βιομηχανίες που σχετίζονται με συστήματα αργής δυναμικής (πχ εργοστάσια χημικών διεργασιών, αλυσίδες παραγωγής κ.α.). Για να εφαρμοστεί ο προβλεπτικός έλεγχος σε νέες επιστημονικές περιοχές υπάρχουν αυξημένες απαιτήσεις όσον αφορά στην απόδοση και την ταχύτητα των αλγορίθμων βελτιστοποίησης που χρησιμοποιούνται και συγκεκριμένα στους αλγόριθμους τετραγωνικού προγραμματισμού.

1.1 Προβλεπτικός έλεγχος (Model predictive Control)

Έστω οτι έχουμε το γραμμικό δυναμικό σύστημα που είναι της μορφής

$$x(t+1) = Ax(t) + Bu(t),\ t = 0,1,\dots \qquad ,x(0) = z \tag{1.1}$$

$$y(t) = Cx(t) \tag{1.2}$$

όπου $u(t) \in U \subseteq R^m$ είναι η είσοδος και $x(t) \in X \subseteq R^n$ είναι οι μεταβλητές κατάστασης και $y(t) \subseteq \mathbb{R}^l$

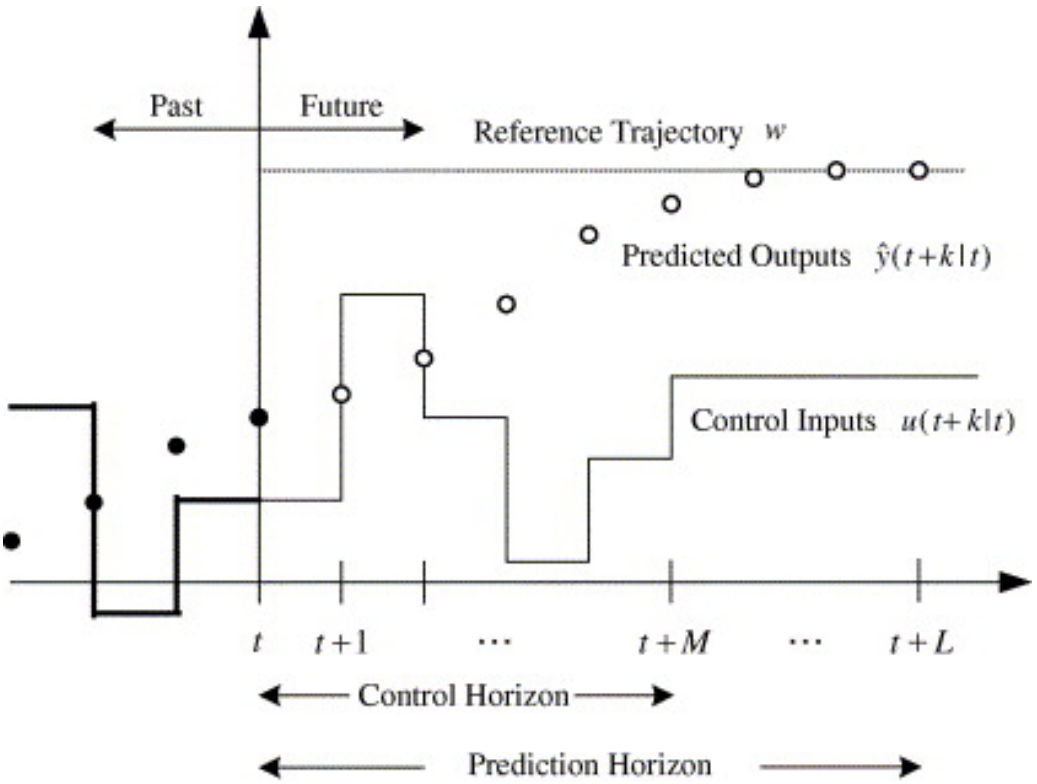
Επίσης ισχύει ότι: $A \in \mathbb{R}^{(n\times n)}, B \in \mathbb{R}^{n\times m}, C \in \mathbb{R}^{l\times n}$

Σε κάθε διακριτή χρονική στιγμή, σκοπός μας είναι η ελαχιστοποίηση μίας αντικειμενικής συνάρτησης που είναι της μορφής:

$$J = \sum_{t=0}^{N} l(x(t), u(t)) \tag{1.3}$$

όπου όμως τα x,u είναι φραγμένα και επίσης πρέπει να ικανοποιούν συνήθως και ένα σύνολο περιορισμών.

Ακολουθεί μια σχηματική απεικόνιση που θα μας βοηθήσει να κατανοήσουμε την ιδέα πίσω από την λειτουργία ενός μοντέλου προβλεπτικού ελέγχου.



Σχήμα 1.: Σχηματική αναπαράσταση της μεθοδολογίας προβλεπτικού ελέγχου

Αν η συνάρτηση $l(x(t), u(t))$ στην αντικειμενική συνάρτηση 1.3 είναι τετραγωνική και οι περιορισμοί γραμμικής μορφής, διαμορφώνεται σε κάθε διακριτή χρονική στιγμή ένα πρόβλημα τετραγωνικού προγραμματισμού (quadratic programming, QP) που υπολογίζει τη βέλτιστη ακολουθία τιμών των μεταβλητών εισόδου. Από αυτές εφαρμόζεται μόνο η πρώτη τιμή (μεταβλητής απόφασης) που υπολογίζεται και μετά την επόμενη χρονική στιγμή t=t+1 υπολογίζουμε ξανά τη βέλτιστη ακολουθία λύνοντας πάλι ένα τετραγωνικό πρόβλημα. Εναλλακτικά μπορεί να εφαρμοστούν οι τιμές που υπολογίστηκαν τη χρονική στιγμή t μέχρι την χρονική στιγμή t+M και μετά να ανανεωθεί το πλάνο ρύθμισης.

Η ρύθμιση _προβλεπτικού _μοντέλου _παρουσιάζει _μία _σειρά _από πλεονεκτήματα _έναντι _άλλων _μεθόδων, από τα οποία τα κυριότερα είναι τα εξής:

- Μπορεί να εφαρμοστεί για τη ρύθμιση πολυμεταβλητών συστημάτων.
- Είναι πολύ χρήσιμη όταν η μελλοντική επιθυμητή συμπεριφορά του συστήματος (π.χ. ρομποτικοί βραχίονες) είναι εκ των προτέρων γνωστή

- Είναι μία εξολοκλήρου ανοιχτή μεθοδολογία βασισμένη σε συγκεκριμένες βασικές αρχές γεγονός που επιτρέπει μελλοντικές επεκτάσεις.
- Λαμβάνει υπόψη τους νεκρούς χρόνους του συστήματος.
- Μπορεί να χρησιμοποιηθεί για την ρύθμιση ενός μεγάλου φάσματος διεργασιών, που παρουσιάζουν είτε σχετικά απλή, είτε ιδιαίτερα πολύπλοκη δυναμική συμπεριφορά, συμπεριλαμβανομένων συστημάτων με μεγάλες _χρονικές καθυστερήσεις ή αστάθειες.
- Είναι ιδιαίτερα ελκυστική για χρήση από προσωπικό με περιορισμένη γνώση προχωρημένης ρύθμισης διότι οι αρχές της είναι απλές και ταυτόχρονα η βαθμονόμηση του _ρυθμιστή _είναι σχετικά εύκολη.

Ωστόσο υπάρχουν, όπως είναι αναμενόμενο και μερικά μειονεκτήματα που αφορούν κυρίως την εφαρμογή τους στην βιομηχανία. Καταρχήν, είναι απαραίτητη η ύπαρξη ενός μοντέλου που να αποδίδει την δυναμική της διεργασίας με τον καλύτερο δυνατό τρόπο. Ο προσδιορισμός θεμελιωδών εξισώσεων, που να βασίζονται στην φυσική του συστήματος, συχνά δεν είναι εύκολος σε μία βιομηχανική μονάδα και επομένως είναι προτιμότερη η ανάπτυξη εμπειρικών μοντέλων, τα οποία ωστόσο πάντα συνοδεύονται από σφάλμα**.** Επίσης, παρόλη την πρόοδο των υπολογιστικών συστημάτων, το πρόβλημα της υπολογιστικής ισχύος που απαιτείται για την επίλυση του προβλήματος βελτιστοποίησης σε πραγματικό χρόνο είναι υπαρκτό. Εκτός του γεγονότος ότι απαιτείται η εύρεση λύσης που να ικανοποιεί τους περιορισμούς στο χρονικό διάστημα που μεσολαβεί μεταξύ δύο διαδοχικών εφαρμογών της μεθόδου, ιδιαίτερη σημασία έχει και η ποιότητα της λύσης για την καλή απόκριση του συστήματος _(Camacho και _Bordons, 1998; Rossiter, 2000).

Αυτό είναι και το κεντρικό αντικείμενο που εξετάζουμε σε αυτή την εργασία, δηλαδή πέραν της διαμόρφωσης της τετραγωνικής αντικειμενικής συνάρτησης προς ελαχιστοποίηση καθώς και τη διαμόρφωση των περιορισμών που διέπουν τις μεταβλητές κατάστασης και εκ-χειρισμού, εξετάζουμε και συγκρίνουμε την

απόδοση (σε χρόνο και ακρίβεια) αλγορίθμων που λύνουν το τετραγωνικό πρόβλημα που προκύπτει από την ανάλυση MPC

1.2 .Κριτήρια επιλογής αλγορίθμων QP για προβλεπτικό έλεγχο:

Οι τεχνικές ελέγχου τύπου MPC θεωρούνται πλέον ώριμες, ενώ έχει αναπτυχθεί κατάλληλη θεωρητική υποδομή που εξασφαλίζει την ευστάθεια και ευρωστία τους. Η ερευνητική κοινότητα έχει στραφεί τα τελευταία χρόνια στην προσπάθεια βελτίωσης της απόδοσης των αλγόριθμων που χρησιμοποιούνται για την επίλυση των προβλημάτων βελτιστοποίησης που διαμορφώνονται σε πραγματικό χρόνο με στόχο να διευρύνουν το πεδίο εφαρμογής τους. Προέκυψε λοιπόν μια σειρά κριτηρίων που ειδικότερα για την περίπτωση ενός αλγόριθμου QP μπορούμε να τα συνοψίσουμε ως εξής:

1. Ταχύτητα ώστε να παρέχει λύση μέσα σε μικρά χρονικά διαστήματα
2. Να μην απαιτείται πολύ μνήμη για να αποθηκευτούν τα δεδομένα του προς βελτιστοποίηση προβλήματος.
3. Να υλοποιείται με απλούς κώδικες προγραμματισμού, ώστε να μπορεί να ενσωματωθεί σε απλές διατάξεις εφαρμογής (microcontroller)
4. Να μπορεί να εκτιμήσει τους χειρότερους δυνατούς αναμενόμενους χρόνους εκτέλεσης ώστε να συμβαδίζει με απαιτήσεις ενός πραγματικού συστήματος σε real-time συνθήκες.

Αυτές οι απαιτήσεις έχουν αποτελέσει το αντικείμενο εκτενούς έρευνας στην κοινότητα MPC κατά την διάρκεια της τελευταίας δεκαετίας. Μέχρι σήμερα πολλοί καλοί αλγόριθμοι και πακέτα για τετραγωνικό προγραμματισμό είναι διαθέσιμοι για επίλυση γραμμικών προβλημάτων MPC όπως για παράδειγμα οι μέθοδοι active-set, μέθοδοι εσωτερικού σημείου (interior point methods) καθώς και η δυαδική ομαλή μέθοδος Newton.

Μία διαφορετική προσέγγιση ακολουθήθηκε στο [Bemporad et al., 2002] όπου προτείνεται μια μεθοδολογία πολυ-παραμετρικού τετραγωνικού προγραμματισμού για την off-line επίλυση του QPπροβλήματος, βάσει της οποίας ο νόμος προβλεπτικού ελέγχου προκύπτει ως μία συνεχής και ανά

στοιχείο γραμμική συνάρτηση του διανύσματος μεταβλητών κατάστασης. Το μεγάλο μειονέκτημα αυτής της μεθόδου είναι ότι περιορίζεται σε σχετικά μικρά προβλήματα (όπως είπαμε, 1 ή δύο εισόδους, μικρούς ορίζοντες , μέχρι 10 μεταβλητές κατάστασης).

Οι συγγραφείς του [Richter et al., 2009, 2011] ήταν οι πρώτοι που εφάρμοσαν την fast-gradient μέθοδο [Nesterov, 1983, 2004] σε περιπτώσεις γραμμικού προβλεπτικού ελέγχου. Στην δημοσίευση [Richter et al., 2009] οι συγγραφείς εφάρμοσαν την πρώτη fast-gradient μέθοδο του Nesterov [Nesterov, 1983] σε ένα πρωτεύων τετραγωνικό πρόβλημα όπου οι μεταβλητές εκ χειρισμού (είσοδοι) βρίσκονται υπο γραμμικούς περιορισμούς όπου όμως αυτοί είναι σύνολα απλής μορφής (πχ –να βρίσκονται εντός κύβου) και κατέληξαν να δώσουν υπολογίσιμα άνω φράγματα για τον αριθμό των επαναλήψεων ανάλογα με το επίπεδο της επιθυμητής υπο-βελτιστοποίησης.

Όσο αφορά στον αλγόριθμο που προτάθηκε από τους Patrinos και Bemporad [2012] για να λύνει προβλήματα MPC τα οποία υποβάλλονται σε γενικούς πολυεδρικούς περιορισμούς στις εισόδους και στις μεταβλητές κατάστασης, αυτός εφαρμόζει την μέθοδο του Nesterov στο δυικό πρόβλημα που προκύπτει. Μάλιστα έχει ρυθμό σύγκλισης της τάξης $O(\frac{1}{\nu^2})$ όπου το ‘’ν’’ είναι ο αριθμός των επαναλήψεων. Επίσης δόθηκαν και τα ανώτατα άνω φράγματα για τον απαιτούμενο αριθμό επαναλήψεων ώστε η λύση να συγκλίνει σε μια δοθείσα ακρίβεια όχι μόνο για τη δυική βελτιστοποίηση αλλά και για την πρωτεύουσα .

2 Ανάλυση Μαθηματικής Βελτιστοποίησης (Mathematical Optimization)

Η γενική μορφή ενός προβλήματος βελτιστοποίησης είναι:

$$\begin{aligned} &minimize && f_0(x) && && (2.1)\\ &subject\ to && f_i(x) \le b_i, && i = 1, \dots\dots, m && \\ & && h_i(x) = 0 && i = 1 \dots . p && \end{aligned}$$

(2.2)

- $x = (x_1, \dots . , x_n)$: *διάνυσμα μεταβλητών που βελτιστοποιεί την* f_0
- $f_0 : \mathbb{R}^n \to \mathbb{R}$: *αντικειμενική συνάρτηση*
- $f_i : \mathbb{R}^n \to \mathbb{R}\ , i = 1, \dots . , m$: *συναρτήσεις ανισοτικών περιορισμών*
- $h_i : \mathbb{R}^n \to \mathbb{R}, i = 1 \dots p$: *περιορισμοί ισοτήτων*

Η (βέλτιστη) λύση x^* δίνει την ελάχιστη δυνατή τιμή για την f_0 ανάμεσα σε όλα τα διανύσματα που ικανοποιούν τους περιορισμούς.

2.1 . Αλγόριθμοι και υπολογιστική πολυπλοκότητα

Η έννοια των αλγορίθμων υπάρχει πολύ πριν την εμφάνιση των μοντέρνων υπολογιστικών συστημάτων. Στην πραγματικότητα, ο άνθρωπος ξεκίνησε τη χρήση και την ανάπτυξη τους ταυτόχρονα με την συστηματική επίλυση προβλημάτων που αντιμετώπιζε. Παρά ταύτα, μετά την εισαγωγή των μοντέρνων υπολογιστικών συστημάτων στα μέσα του περασμένου αιώνα, έγινε πολύ δημοφιλής η αναφορά των αλγορίθμων σαν προγράμματα ηλεκτρονικού υπολογιστή. Έτσι, σήμερα, σαν αλγόριθμους χαρακτηρίζουμε την περιγραφή υψηλού επιπέδου ενός προγράμματος ηλεκτρονικού υπολογιστή, που είναι με τη σειρά της μια βήμα-προς-βήμα προδιαγραφή της διαδικασίας επίλυσης ενός συγκεκριμένου προβλήματος.

Κάθε βήμα ενός αλγορίθμου αποτελείται από ένα πεπερασμένο αριθμό λειτουργιών, οι οποίες γενικά περιλαμβάνουν αριθμητικές πράξεις, λογικές

συγκρίσεις, διαδικασίες ελέγχου και λειτουργίες αποθήκευσης και ανάκτησης δεδομένων από τη μνήμη και τις περιφερειακές μονάδες του ηλεκτρονικού υπολογιστή.

Ας υποθέσουμε ότι ο αλγόριθμος χρησιμοποιείται για την επίλυση ενός συγκεκριμένου προβλήματος . Είναι πολύ λογικό να υποθέσουμε ότι για κάποια σειρά από υποδείγματα μεγάλου μεγέθους, ο αλγόριθμος θα χρειαστεί για την επίλυση του προβλήματος σημαντικό υπολογιστικό χρόνο. Πρέπει να αναφέρουμε εδώ ότι το μέγεθος ενός αντικειμένου, όπως αυτό του υποδείγματος ενός προβλήματος αριστοποίησης, είναι κάτι που δεν μπορεί να οριστεί μονοσήμαντα, και σε όλες τις περιπτώσεις είναι συνάρτηση του τρόπου με τον οποίο το συγκεκριμένο αντικείμενο έχει αποθηκευτεί στη μνήμη του ηλεκτρονικού υπολογιστή. Στη γενικότερη περίπτωση, σαν μέγεθος ενός αντικειμένου μπορεί κάποιος θεωρήσει την έκταση που καταλαμβάνει στον αποθηκευτικό χώρο του ηλεκτρονικού υπολογιστή. Έτσι, η αποτίμηση της απόδοσης του αλγορίθμου θα εξεταστεί σε όρους μεγέθους του υποδείγματος του προβλήματος.

Είναι γενικά δύσκολο και αδόκιμο το να χρησιμοποιηθεί για την αποτίμηση της απόδοσης ενός αλγορίθμου ο ακριβής αριθμός των βασικών λειτουργιών που ο αλγόριθμος εκτελεί στην προσπάθεια του να βρει την βέλτιστη λύση ενός προβλήματος αριστοποίησης. Υπάρχουν αρκετοί λόγοι για αυτό. Η ανομοιογένεια των χαρακτηριστικών των γλωσσών προγραμματισμού και των μεταφραστών τους σε εκτελέσιμο κώδικα είναι ένας σημαντικός λόγος. Η διαφορετική συμπεριφορά ως προς τον τύπο των δεδομένων σε κάθε μια από τις βασικές λειτουργίες του κώδικα που θα χρησιμοποιηθεί είναι ένας άλλος. Στη γενικότερη περίπτωση, είναι αρκετή μια κατά προσεγγιστική περιγραφή της πολυπλοκότητας, δηλαδή των απαιτούμενων υπολογιστικών πόρων για την εκτέλεση του αλγορίθμου, που να είναι όμως κοινός για τον τρόπο περιγραφής όλων των αλγορίθμων.

Είναι καθιερωμένο σήμερα στην επιστήμη των ηλεκτρονικών υπολογιστών η

χρήση ασυμπτωτικών ορίων (φραγμάτων) για τη μέτρηση των υπολογιστικών πόρων που είναι απαραίτητοι σε κάποιον αλγόριθμο που επιλύει κάποιο υπολογιστικό πρόβλημα. Δεδομένης μιας συνάρτησης f(n) που απεικονίζει ακεραίους σε πραγματικούς αριθμούς, η χρήση των παρακάτω συμβόλων είναι ουσιαστική για την ανάλυση ασυμπτωτικών ορίων της πολυπλοκότητας αλγορίθμων για την επίλυση μαθηματικών προβλημάτων:

- **O(f(n)):** η κλάση των συναρτήσεων που είναι τέτοιες ώστε για μια συνάρτηση g από αυτές, πάντα να υπάρχει κάποιος σταθερός αριθμός cg έτσι ώστε $f(n) \geq c$g(n) για όλες τις πεπερασμένες τιμές του n. Στην περίπτωση αυτή, η κλάση Ot(n)) είναι όλες οι συναρτήσεις που είναι το πολύ όσο και οι συναρτήσεις t(n).
- **o(f(n)):** η κλάση των συναρτήσεων που είναι τέτοιες ώστε για μια συνάρτηση g από αυτές, να ισχύει $\lim_{n\to\infty} \frac{g(n)}{f(n)}$,=0, για όλες τις πεπερασμένες τιμές του n. Στην περίπτωση αυτή, η κλάση ot(n)) είναι όλες οι συναρτήσεις που είναι μικρότερες των συναρτήσεων t(n).
- **Ω(f(n)):** η κλάση των συναρτήσεων που είναι τέτοιες ώστε για μια συνάρτηση g από αυτές, πάντα να υπάρχει κάποιος σταθερός αριθμός c_g έτσι ώστε $f(n) \leq c_g g(n)$ για όλες τις πεπερασμένες τιμές του n. Στην περίπτωση αυτή, η κλάση Ω(f(n)) είναι όλες οι συναρτήσεις που είναι τουλάχιστον όσο και οι συναρτήσεις f(n).
- **ω(f(n))**: η κλάση των συναρτήσεων που είναι τέτοιες ώστε για μια συνάρτηση g από αυτές, να ισχύει
  $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0,$
  για όλες τις πεπερασμένες τιμές του n. Στην περίπτωση αυτή, η κλάση ω(f(n)) είναι όλες οι συναρτήσεις που είναι μεγαλύτερες των συναρτήσεων t(n).
- **Θ(f(n))**: η κλάση των συναρτήσεων που είναι τέτοιες ώστε για μια συνάρτηση g από αυτές, να ισχύει g(n) = O(f(n)) και g(n) = Ω(f(n)) , για όλες τις πεπερασμένες τιμές του n. Στην

περίπτωση αυτή, η κλάση Θ(f(n)) είναι όλες οι συναρτήσεις που είναι της ίδιας τάξης μεγέθους με τις συναρτήσεις f(n).

Ο χρόνος εκτέλεσης ενός αλγορίθμου για δεδομένο υπόδειγμα συγκεκριμένου προβλήματος ορίζεται σαν ο αριθμός των βασικών λειτουργιών που διεξάγονται κατά τη διάρκεια εκτέλεσης του αλγορίθμου για το συγκεκριμένο υπόδειγμα. Έστω ένας αλγόριθμος Α ο οποίος λύνει ένα πρόβλημα αριστοποίησης Q και έστω f(n) μια συνάρτηση. Η χρονική πολυπλοκότητα του αλγορίθμου Α είναι O(f(n)) εάν υπάρχει μια συνάρτηση $g(n) \in O(f(n))$ τέτοια ώστε για κάθε n ≥ 0 , ο χρόνος εκτέλεσης του Α να περιορίζεται από την g(n) για όλες τις δυνατές τιμές του n.

Ένας αλγόριθμος Α είναι αλγόριθμος πολυωνυμικού χρόνου εάν υπάρχει μια σταθερά c τέτοια ώστε η χρονική πολυπλοκότητα του αλγορίθμου Α να είναι $O(n^c)$. Ένα πρόβλημα αριστοποίησης μπορεί να λυθεί σε πολυωνυμικό χρόνο αν υπάρχει αλγόριθμος πολυωνυμικού χρόνου που να μπορεί να το λύνει.
Οι αλγόριθμοι πολυωνυμικού χρόνου θεωρούνται εύκολες και εφικτές υπολογιστικές διαδικασίες. Η διαφορά της απόδοσης των πολυωνυμικών αλγορίθμων σε σχέση με αυτούς εκθετικής πολυπλοκότητας δίδεται στα δεδομένα των Πινάκων 1 και 2.

Πίνακας 1. Υπολογιστικοί χρόνοι προβλημάτων πολυωνυμικής και εκθετικής πολυπλοκότητας

Πολυπλοκότητα	Μέγεθος υποδείγματος προβλήματος (n)				
	10	20	30	40	50
$O(n)$	0.00001 s	0.00002 s	0.00003 s	0.00004 s	0.00005 s
$O(n^2)$	0.0001 s	0.0004 s	0.0009 s	0.0016 s	0.0025 s
$O(n^3)$	0.001 s	0.008 s	0.027 s	0.064 s	0.125 s
$O(2^n)$	0.001 s	1 s	17.9 min	12.7 days	35.7 years

Πίνακας 2. Επίδραση αύξησης υπολογιστικής δύναμης (μέγεθος προβλήματος που μπορεί να επιλυθεί σε 1 h)

Πολυπλοκότητα	Σημερινή Υπολογιστική Ισχύς	100πλάσια Υπολογιστική Ισχύς	1000πλάσια Υπολογιστική Ισχύς
$O(n)$	n_1	$100n_1$	$1000n_1$
$O(n^2)$	n_2	$10n_2$	$31.6n_2$
$O(n^3)$	n_3	$4.64n_3$	$10n_3$
$O(2^n)$	n_4	$n_4+6.64$	$n_4+9.97$

Στη γενικότερη περίπτωση, δεδομένου κάποιου προβλήματος αριστοποίησης, ο εντοπισμός κάποιου αλγόριθμου πολυωνυμικού χρόνου για την επίλυση του είναι πολύ σημαντική διαδικασία.

2.2. Θεωρία πολυπλοκότητας

Η θεωρία πολυπλοκότητας παίζει έναν ουσιαστικό ρόλο στην ανάλυση των προβλημάτων αριστοποίησης. Η θεωρία αυτή αναπτύχθηκε από τη μελέτη αντίστοιχων υπολογιστικών προβλημάτων, με την ελπίδα ότι θα μπορούσε να παράξει αξιόπιστα κάτω φράγματα στην υπολογιστική πολυπλοκότητα κάποιων προβλημάτων αριστοποίησης. Μια σημαντική έννοια της θεωρίας αυτής, στην οποία καταλήγουν με τον ένα ή τον άλλο τρόπο τα περισσότερα προβλήματα αριστοποίησης που έχουν πρακτικό ενδιαφέρον, είναι αυτή των προβλημάτων απόφασης. Ένα πρόβλημα απόφασης είναι κάποιο πρόβλημα για το οποίο για όλα τα εμπλεκόμενα υποδείγματα μπορεί να υπάρχει σαν απάντηση του προβλήματος μία μόνον από τις εξής δύο δυνατές λογικές καταστάσεις: να έχει λύση (εφικτό) ή να μην έχει λύση (μη εφικτό).

Ένα πρόβλημα αριστοποίησης μπορεί να μετασχηματιστεί σε ένα πρόβλημα

απόφασης με την εισαγωγή μιας παραμέτρου η οποία μπορεί να χρησιμοποιηθεί για σύγκριση με την άριστη τιμή ενός συγκεκριμένου υποδείγματος. Για παράδειγμα, η εκδοχή προβλήματος απόφασης για το πρόβλημα του Περιοδεύοντος Πωλητή μπορεί να διατυπωθεί ως εξής: Το υπόδειγμα ενός προβλήματος απόφασης για το πρόβλημα αυτό είναι της μορφής (G,k), όπου G είναι ο πίνακας του κόστους μετακινήσεων μεταξύ των εμπλεκομένων πόλεων και k ένας ακέραιος αριθμός. Η ερώτηση που πρέπει να απαντηθεί για δεδομένο υπόδειγμα (G,k) είναι η εξής: «Υπάρχει μια διαδρομή για τον πωλητή που επισκέπτεται όλες τις εμπλεκόμενες πόλεις ακριβώς μια φορά και επιστρέφει σε αυτήν από την οποία ξεκίνησε και να έχει συνολικό κόστος μετακινήσεων φραγμένο από τον αριθμό k;

Ένας αλγόριθμος A επιλύει ένα πρόβλημα απόφασης Q εάν για κάθε εφικτό υπόδειγμα του προβλήματος ο αλγόριθμος επιστρέφει σαν αποτέλεσμα μια καταφατική λογική κατάσταση, ενώ για κάθε μη εφικτό υπόδειγμα του προβλήματος ο αλγόριθμος επιστρέφει σαν αποτέλεσμα μια αρνητική λογική κατάσταση. Ένα πρόβλημα απόφασης Q ανήκει στην κλάση P εάν μπορεί να επιλυθεί από έναν αλγόριθμο πολυωνυμικού χρόνου.

Κατά μία πιο γενική και εκτεταμένη έννοια, είναι αποδεκτό ότι ένα πρόβλημα αριστοποίησης Q ανήκει στην κλάση προβλημάτων P αν μπορεί να λυθεί από κάποιον αλγόριθμο σε πολυωνυμικό χρόνο, ακόμα και αν το πρόβλημα αυτό δεν είναι εύκολο να διατυπωθεί σαν πρόβλημα απόφασης. Δυστυχώς, πολλά προβλήματα αποφάσεων που προκύπτουν από προβλήματα αριστοποίησης δεν δείχνουν να ανήκουν στην κλάση P, δεν έχουν βρεθεί δηλαδή αλγόριθμοι πολυωνυμικού χρόνου για να τα επιλύσουν. Μεγάλος αριθμός όμως από τα προβλήματα αυτά μπορεί να συνδεθεί με πολυωνυμικούς αλγορίθμους επίλυσης χρησιμοποιώντας την έννοια της κλάσης των non-deterministic polynomial (NP) αλγόριθμων:

Έτσι, ένα πρόβλημα Q που ανήκει στην κλάση NP είναι ένα πρόβλημα στο οποίο μια λύση σε ένα εφικτό του υπόδειγμα, μπορεί να επαληθευτεί σε πολυωνυμικό

χρόνο από τον αλγόριθμο Α. Το πρόβλημα απόφασης για την περίπτωση του Περιοδεύοντος Πωλητή, για παράδειγμα, ανήκει στην κλάση NP.

Είναι προφανές ότι η κλάση P είναι υποκλάση της NP. Μια σημαντική έννοια της θεωρίας της NP-πολυπλοκότητας είναι και η έννοια της υποβαθμισιμότητας που παρουσιάζεται παρακάτω:

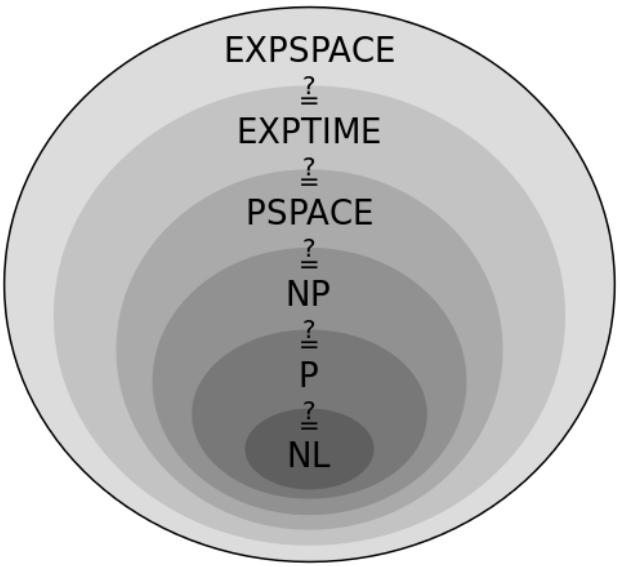
Θεωρούμε δύο προβλήματα απόφασης Q1 και Q2. Το πρόβλημα Q1 ονομάζεται πολυωνυμικά υποβαθμίσιμο στο πρόβλημα Q2 (συμβολίζεται σαν $Q_1 \leq_m^p Q_2$) τότε και μόνον τότε όταν υπάρχει μια συνάρτηση r υπολογιζόμενη σε πολυωνυμικό χρόνο, έτσι ώστε αν x είναι κάποιο εφικτό υπόδειγμα του Q1, το υπόδειγμα r(x) είναι εφικτό υπόδειγμα του προβλήματος Q2.
Η σχέση $Q_1 \leq_m^p Q_2$ υποδηλώνει ότι το πρόβλημα Q2 δεν είναι ευκολότερο από το πρόβλημα Q1 (ισοδύναμα, το πρόβλημα Q1 δεν είναι δυσκολότερο από το πρόβλημα Q2). Έτσι, η σχέση $Q_1 \leq_m^p Q_2$ θέτει ένα κάτω φράγμα για την υπολογιστική πολυπλοκότητα του προβλήματος Q2 σε όρους του προβλήματος Q1 και ισοδύναμα θέτει ένα άνω φράγμα για την υπολογιστική πολυπλοκότητα του προβλήματος Q1 σε όρους του προβλήματος Q2. Συγκεκριμμένα, ισχύει το παρακάτω: Θεωρούμε δύο προβλήματα απόφασης Q1 και Q2. Εάν ισχύει $Q_1 \leq_m^p Q_2$ και το πρόβλημα Q2 ανήκει στην κλάση P, τότε το πρόβλημα Q1 ανήκει επίσης στην κλάση P. Ακόμη περισσότερο, αν υπάρχει κάποιος πολυωνυμικός αλγόριθμος που να επιλύει ένα πρόβλημα μιας συγκεκριμένης κλάσης, τότε θα υπάρχει και κάποιος αντίστοιχος πολυωνυμικός αλγόριθμος που να επιλύει οποιοδήποτε πρόβλημα της ίδιας κλάσης, αρκεί αυτός να βρεθεί.

Ένα πρόβλημα απόφασης Q είναι NP-κοπιώδες όταν όλα τα προβλήματα της κλάσης NP είναι πολυωνυμικά υποβαθμίσιμα στο Q. Ένα πρόβλημα απόφασης Q είναι NP-ακέραιο όταν είναι NP-κοπιώδες και ανήκει στην κλάση NP.
Για τα προβλήματα αυτά ισχύουν τα ακόλουθα:

Θεωρούμε τρία προβλήματα απόφασης Q1, Q2 και Q3. Εάν $Q_1 \leq_m^p Q_2$ και $Q_2 \leq_m^p Q_3$ τότε $Q_1 \leq_m^p Q_3$. Θεωρούμε δύο προβλήματα απόφασης Q1 και Q2. Εάν

το πρόβλημα Q1είναι NP-κοπιώδες και ισχύει $Q_1 \leq_m^p Q_2$, τότε το πρόβλημα Q2 είναι NP-κοπιώδες επίσης.

Οι έννοιες που κρύβονται πίσω από τα παραπάνω θεωρήματα διαμορφώνουν ένα εξαιρετικά σημαντικό σύστημα εργασίας για την απόδειξη της υπολογιστικής δυσκολίας επίλυσης των προβλημάτων αριστοποίησης. Αν κάποιος θέλει να αποδείξει ότι κάποιο πρόβλημα Q είναι NP-κοπιώδες, αρκεί να επιλέξει ένα ήδη γνωστό NP-κοπιώδες πρόβλημα Q' και να δείξει ότι ισχύει $Q' \leq_m^p Q$. Αν επιτύχει, τότε το πρόβλημα Q' είναι NP-κοπιώδες και κατά συνέπεια είναι απίθανο να λύνεται από κάποιον αλγόριθμο πολυωνυμικού χρόνου. Επιπρόσθετα, το πρόβλημα αυτό προστίθεται στο σύνολο των ήδη γνωστών NP-κοπιωδών προβλημάτων και μπορεί να αποδειχθεί χρήσιμο για αντίστοιχες αποδείξεις σε άλλα προβλήματα. Τις τελευταίες δεκαετίες, χρησιμοποιήθηκε ευρέως η τεχνική αυτή για την εξέταση χιλιάδων προβλημάτων αριστοποίησης ως NP-κοπιώδη. Έτσι, θεωρείται ότι όλες αυτές οι χιλιάδες προβλημάτων είναι αδύνατο να λυθούν σε πολυωνυμικό χρόνο. Αντίθετα, αν ένα από αυτά λυθεί σε πολυωνυμικό χρόνο, τότε θα έχουν λυθεί όλα. Βέβαια, το σύστημα αυτό βασίζεται σήμερα στην παρακάτω εικασία: P ≠ NP , δηλαδή υπάρχουν προβλήματα που να ανήκουν στη κλάση NP που δεν μπορούν να λυθούν σε πολυωνυμικό χρόνο. Μέχρι σήμερα δεν υπάρχει απόδειξη για την εικασία αυτή.



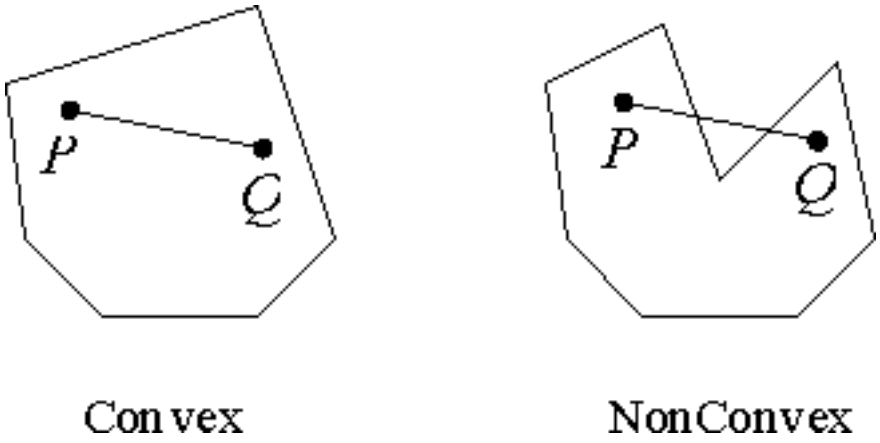
Σχήμα 2: Αναπαράσταση της σχέσης των κλάσεων όπου η μία είναι υποσύνολο της άλλης.

2.3. Στοιχεία απο την Κυρτή Ανάλυση:

Στην ανάλυση που θα ακολουθήσει θα θεωρούμε δεδομένο ότι η αντικειμενική συνάρτηση καθώς και οι περιορισμοί θα είναι κυρτοί. Μια συνάρτηση (ή ένας περιορισμός) λέγεται κυρτός όταν το πεδίο ορισμού είναι κυρτό σύνολο και όταν ισχύει η σχέση:

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta)f(x_2)\,, \forall (x_1, x_2) \in Domf, \theta \in [0,1] \quad (2.3.1)$$

Στην περίπτωση οπού η παραπάνω σχέση ισχύει με γνήσια ανισότητα για $\theta \in (0,1) και\ x_1 \neq x_2$ τότε η συνάρτηση λέγεται αυστηρά κυρτή.



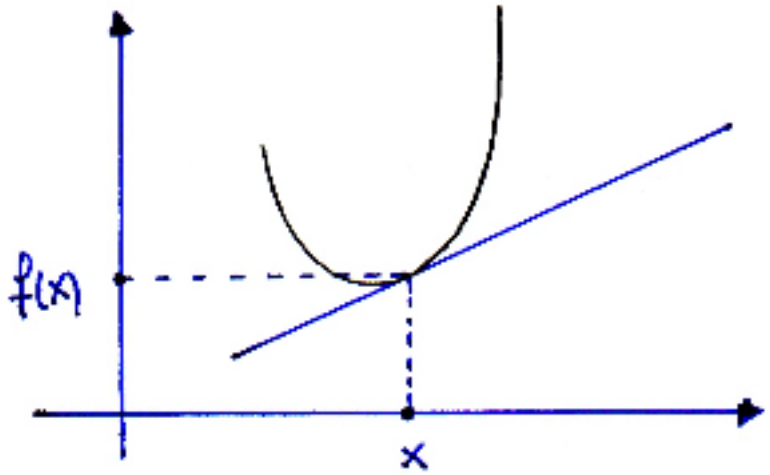
Σχήμα 3: Ένα κυρτό και ένα μη κυρτό σύνολο.

1η συνθήκη για εξασφάλιση κυρτότητας:

Έστω $f: \Omega \to \mathbb{R}, \Omega \subseteq \mathbb{R}^n$. Η f θα είναι κυρτή αν το Ω είναι κυρτό και αν $\forall (x, y) \in \Omega$ ισχύει ότι

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \quad \text{(Προσέγγιση Taylor } 1^{ου} \text{ βαθμού)} \quad (2.3.2)$$

δηλαδή όπως βλέπουμε στο σχήμα:



Σχήμα 4: Η μαύρη καμπύλη απεικονίζει την f(y) ενώ η μπλέ απεικονίζει την προσέγγιση Taylor 1ου βαθμού της f

Επομένως αρκεί να βρούμε ένα σημείο $x^* \in dom(f)$ τέτοιο ώστε $\nabla f(x^*) = 0$ γιατί τότε θα έχουμε $f(y) \geq f(x^*), \forall y \in \Omega$ άρα το x^* είναι ολικό ελάχιστο.

2η συνθήκη εξασφάλισης κυρτότητας:

Η συνάρτηση f θα είναι κυρτή, αν το πεδίο ορισμού της είναι κυρτό σύνολο , και αν το μητρώο Hessian είναι θετικά ημιορισμένο δηλαδή αν:

$$\nabla^2 f(x) \geq 0 \tag{2.3.3}$$

Παρατήρηση: Ο πίνακας αυτός είναι τετραγωνικός και συμμετρικός επομένως εάν είναι θετικά ημιορισμένος, τότε έχει μη αρνητικές ιδιοτιμές. Αντίστοιχα εάν είναι θετικά ορισμένος (δηλαδή $\nabla^2 f(x) \in \mathbb{S}^n_{++}$) τότε όλες οι ιδιοτιμές θα είναι θετικές.

Αν ισχύει ότι $\nabla^2 f(x) \leq 0$ τότε η συνάρτηση θα είναι κοίλη και σε αυτή τη περίπτωση τα κρίσιμα σημεία δίνουν μέγιστο

Θα αναφερθούμε στη δομή και λύση δύο ειδών προβλημάτων βελτιστοποίησης.

1. Αυτά όπου η αντικειμενική συνάρτηση καθώς και οι περιορισμοί είναι γραμμικοί (άρα και κυρτοί) (L.P)
2. Η αντικειμενική συνάρτηση είναι τετραγωνική αλλά κυρτή και οι περιορισμοί γραμμικοί (άρα και αυτοί δημιουργούν ένα κυρτό σύνολο.) (Q.P)

Πριν αναφερθούμε στην θεωρία που δείχνει πως να βρίσκουμε τα βέλτιστα σε κάθε μια από τις παραπάνω περιπτώσεις, αξίζει να σημειωθεί το γεγονός ότι η αντικειμενική συνάρτηση είναι κυρτή, μας εξασφαλίζει την ύπαρξη ολικού ελάχιστου. Επομένως, αν βρούμε ένα κρίσιμο σημείο που δίνεται από την κλίση ∇f, αυτό εξαιτίας τις κυρτότητας δεν αποτελεί τοπικό βέλτιστο αλλά ολικό.

2.4. Δυϊκότητα:

Στην δυϊκότητα χειριζόμαστε τους περιορισμούς με τρόπο ώστε να εισέλθουν στην αντικειμενική συνάρτηση.

Έστω οτι έχουμε ένα πρόβλημα της μορφής:

$$\begin{aligned} &minimize && f_0(x) \ , x \in \boldsymbol{D} \subseteq \mathbb{R}^n \\ &subject\ to && f_i(x) \leq 0, \qquad i = 1, \dots \dots \dots, m \\ & && h_i(x) = 0 \qquad i = 1 \dots . p \end{aligned}$$

όπου δεν απαιτείται η f να είναι κυρτή, και έστω p^*η βέλτιστη λύση.

Ορίζουμε την συνάρτηση.

$$Lagrangian: L: \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}, με\ dom(L) = D \times \mathbb{R}^m \times \mathbb{R}^p$$

$$L(x, \lambda, \nu) = f(x_0) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \qquad (2.4.1)$$

όπου αποτελεί έναν σταθμισμένο άθροισμα της συνάρτησης και των περιορισμών. Οι συντελεστές λ_i αποτελούν τους πολλαπλασιαστές Lagrange που σχετίζονται με τις σχέσεις $f_i(x) \leq 0$ ενώ οι συντελεστές ν_i είναι πολλαπλασιαστές Lagrange που σχετίζονται με τις συναρτήσεις $h_I(x) = 0$. Αυτοί οι συντελεστές πολλές φορές συναντώνται και με την ονομασία «δυϊκές» μεταβλητές. Μία σημαντική παρατήρηση είναι πως αφού βρούμε τις τιμές των συντελεστών, οι συναρτήσεις $f_i(x)$ μπορούν να λάβουν θετικές τιμές και τότε οι

συντελεστές αυτοί θα έχουν τον ρόλο της ποινής για κάθε μονάδα που ξεπερνιέται ο περιορισμός.

Η δυϊκή μορφή της συνάρτησης Lagrange είναι η ακόλουθη:

$$g(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu) \tag{2.4.2}$$

η οποία είναι κοίλη.

Ιδιότητα κάτω φράγματος:

Αν $\lambda \geq 0, \tau\acute{o}\tau\varepsilon\ g(\lambda, \nu) \leq p^*$

Απόδειξη: Αν το $\hat{x}$ είναι εφικτό σημείο και $\lambda \geq 0$ τότε:

$$f_0(\hat{x}) \geq L(\hat{x}, \lambda, \nu) \geq \inf_{x \in D} L(x, \lambda, \nu) = g(\lambda, \nu)$$

Άρα ελαχιστοποιώντας σε όλο το πεδίο των εφικτών $\hat{x}$

δίνει: $p^* \geq g(\lambda, \nu)$

Παράδειγμα εφαρμογής:

$$minimize\ x^T x$$

$$subject\ to\ Ax = b$$

Η δυϊκή συνάρτηση σύμφωνα με τα παραπάνω θα είναι η:

$$L(x, \nu) = x^T x + \nu^T (Ax - b)$$

Για να ελαχιστοποιήσουμε την L ως προς x , θέτουμε την κλίση ίση με 0:

$$\nabla_x L(x, \nu) = 2x + A^T \nu = 0$$

Αντικαθιστώντας στην L παίρνουμε:

$$g(\nu) = L\left(-\frac{1}{2} A^T \nu, \nu\right) = -\frac{1}{4} \nu^T A A^T \nu - b^T \nu$$

και αυτή είναι η δυϊκή συνάρτηση η οποία είναι κοίλη ως προς ν.

Σύμφωνα με την ιδιότητα του κάτω φράγματος θα έχουμε:

$$p^* \geq -\frac{1}{4} \nu^T A A^T \nu - b^T \nu\ \gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ \nu$$

Τελικά το δυϊκό πρόβλημα που βρίσκει το καλύτερο κάτω φράγμα για την τιμή της αντικειμενικής συνάρτησης στο πρωτεύον πρόβλημα είναι το:

$$maximize\ g(\lambda, v)$$
$$subject\ to\ \lambda \geq 0$$

το οποίο είναι κυρτό πρόβλημα βελτιστοποίησης με τιμή d^*.

Έτσι αν έχουμε ένα γραμμικό πρόβλημα σε κανονική μορφή, το (Lagrangian)δυϊκό του θα είναι:

Primal: $minimize\ c^T x$ $\quad$ $subject\ to\ Ax = b\ , x \geq 0$

Dual: $maximize\ \ -b^T v$ $\quad$ $subject\ to\ A^T v + c \geq 0$

Αδύναμη και ισχυρή δυϊκότητα (weak-strong duality):

Θα λέμε ότι σε ένα πρόβλημα έχουμε αδύναμη δυϊκότητα όταν θα ισχύει:

$$d^* \leq p^* \tag{2.4.3}$$

Προφανώς ,όπως δείξαμε παραπάνω ,αυτή η ανισότητα θα ισχύει πάντοτε είτε έχουμε κυρτά είτε μη κυρτά προβλήματα.

Σημείωση: η διαφορά $p^* - d^* \geq 0$ ονομάζεται δυϊκό κενό (duality gap)

Αντίστοιχα, λέμε ότι έχουμε ισχυρή δυϊκότητα (strong duality) όταν ισχύει η ισότητα: $d^* = p^*$ η οποία συνήθως ισχύει για κυρτά προβλήματα.

2.5. Κριτήριο του Slater για εξασφάλιση ισχυρής δυϊκότητας:

Για ένα κυρτό πρόβλημα της μορφής: $minimize\ f_0(x)$

$$subject\ to\ f_i(x) \leq 0, \quad i = 1, \dots, m$$
$$Ax = b$$

θα ισχύει ισχυρή δυϊκότητα αν είναι αυστηρά εφαρμόσιμο δηλαδή:

$$\exists x \in intD: \quad f_i(x) < 0, \quad i = 1, \dots \dots, m \tag{2.4.4}$$

Το παραπάνω κριτήριο εξασφαλίζει και την εύρεση του δυϊκού βέλτιστου εάν $p^* > -\infty$.

2.6. Karush-Kuhn-Tucker συνθήκες:

Οι ακόλουθες συνθήκες ονομάζονται ΚΚΤ και αφορούν ένα πρόβλημα με διαφορίσιμες f_i, h_i

- Ισχύς των πρωτευόντων περιορισμών
  $f_i(x) \leq 0, i = 1{:}\, m, h_i(x) = 0, i = 1{:}\, p$
- Δυϊκοί περιορισμοί: $\lambda \geq 0$
- Συμπληρωματικές σκιώδεις τιμές: $\lambda_i f_i(x) = 0, \forall i \in [1, m]$
- Η κλίση της συνάρτησης Lagrange ως προς χ να είναι 0 δηλαδή

$$\nabla_x L = 0 \leftrightarrow \nabla f_0(x) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x) + \sum_{i=1}^{p} v_i \nabla h_i(x) = 0 \tag{2.4.5}$$

Αν ισχύει η ισχυρή δυϊκότητα και τα x,λ,ν είναι βέλτιστα, τότε πρέπει να ικανοποιούν τις συνθήκες ΚΚΤ.

Για ένα κυρτό πρόβλημα, οι παραπάνω συνθήκες είναι αναγκαίες και ικανές για εξασφάλιση ότι τα $\bar{x}\bar{\lambda}\bar{v}$ είναι βέλτιστα, αν τις ικανοποιούν.

3. Γραμμικός Προγραμματισμός:

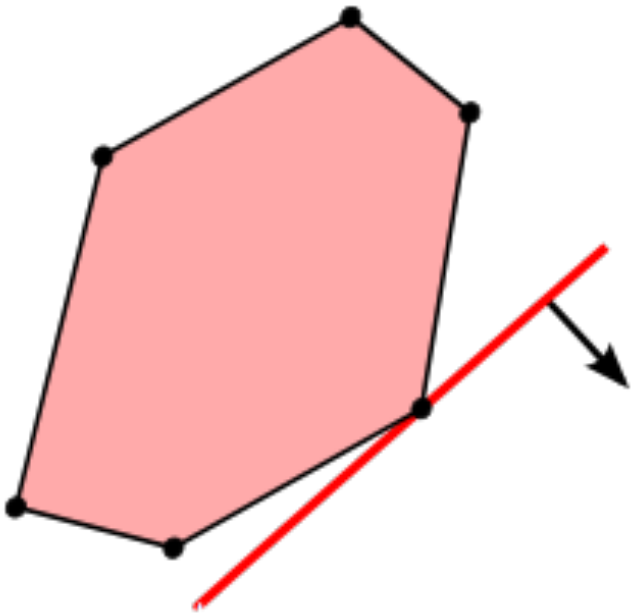
3.1. Εισαγωγή

Δομή: $minimize: c^T x + d$

$subject\ to: Gx \leq h$

Εδώ το κυρτό πρόβλημα περιλαμβάνει μια αντικειμενική συνάρτηση γραμμική όπως και οι περιορισμοί. Το σύνολο όπου είναι εφικτή η λύση είναι ένα πολύεδρο όπως φαίνεται στο σχήμα (LP-feasible domain):

Στο Σχήμα 4 έχουμε μια αναπαράσταση ενός γραμμικού προβλήματος δύο μεταβλητών με έξι περιορισμούς. Το σύνολο των εφικτών λύσεων είναι το κόκκινο πεδίο και σχηματίζει ένα πολύγωνο (ένα πολύτοπο δύο διαστάσεων). Η κόκκινη γραμμή απεικονίζει την ισοσταθμική της γραμμικής αντικειμενικής συνάρτησης (προφανώς οι ισοσταθμικές μίας γραμμικής ν-διάστατης συνάρτησης είναι υπερ-επίπεδα) ενώ το βέλος δείχνει προς την κατεύθυνση που βελτιστοποιούμε (δηλαδή προς την αντίθετη κατεύθυνη του gradient). Θα δούμε παρακάτω ότι σύμφωνα με την γενίκευση των πολλαπλασιαστών Lagrange (ΚΚΤ συνθήκες) έχουμε βέλτιστο στα σημεία όπου η κλίση της συνάρτησης δείχνει προς την ίδια κατεύθυνση με την κλίση των περιορισμών.



(Σχήμα 5: Γραμμικός Προγραμματισμός)

Πρέπει να αναφέρουμε οτι σε αυτή την κατηγορία προβλημάτων (γραμμικά) η βέλτιστη λύση βρίσκεται πάντοτε σε κάποια κορυφή του πολύεδρου. Αντίθετα στα τετραγωνικά προβλήματα η λύση μπορεί να είναι και στην πλευρά του πολυέδρου.

3.2. Αλγόριθμοι επίλυσης Προβλημάτων Γραμμικού Προγραμματισμού:

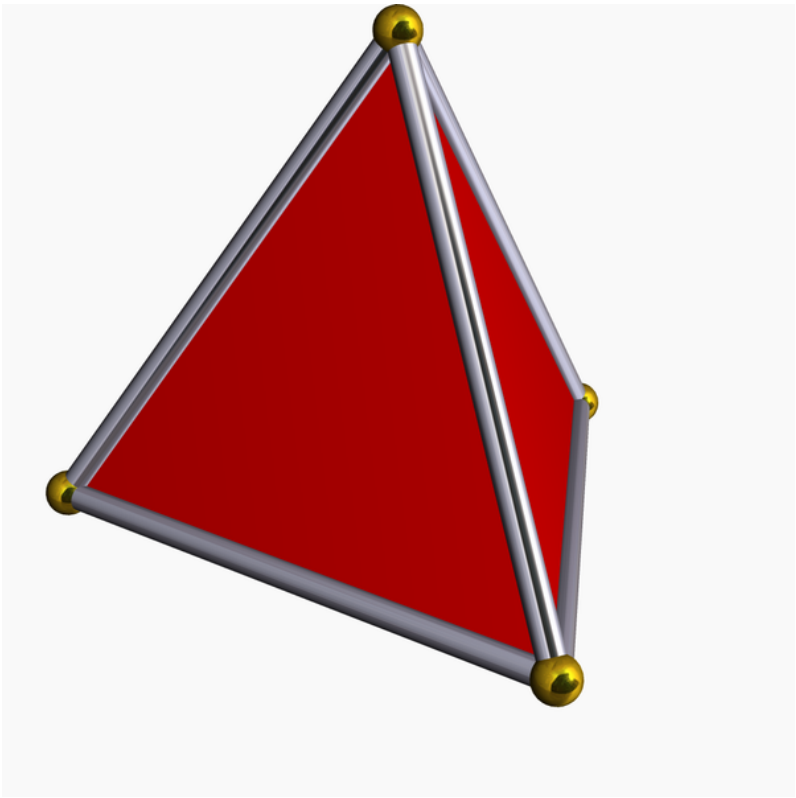
3.2.1 Μέθοδος-Simplex:

Στην μαθηματική βελτιστοποίηση , ο αλγόριθμος του Dantzig είναι ένας διάσημος αλγόριθμος για γραμμικό προγραμματισμό. Το όνομα του αλγορίθμου προέρχεται από την έννοια του simplex όπως προτάθηκε από τον T.S Motzkin. Στην γεωμετρία, ένα simplex είναι η γενίκευση της έννοιας του τριγώνου ή τετραέδρου σε οποιαδήποτε διάσταση. Συγκεκριμένα, ένα k-simplex είναι ένα κ-διάστατο πολύτοπο που αποτελεί την κυρτή θήκη των κ+1 κορυφών του.

Φορμαλιστικά έχουμε:
Έστω $u_0, \dots, u_k \in \mathbb{R}^n γραμμικώς\ ανεξάρτητα\ σημεία$. Τότε το simplex χωρίο που καθορίζεται από αυτά είναι το σύνολο των σημείων για τα οποία έχουμε:

$$C = \{\theta_0 u_0 + .. + \theta_k u_k | \theta_i \geq 0, 0 \leq i \leq k, \sum_{i=0}^{k} \theta_i = 1\}$$

Για παράδειγμα, ένα 2-simplex είναι ένα τρίγωνο, ένα 3-simplex είναι ένα τετράεδρο ενώ ένα 0-simplex είναι ένα σημείο και ένα ευθύγραμμο τμήμα θεωρείται 1-simplex. Ένα simplex δηλαδή ορίζεται σαν το μικρότερο κυρτό σύνολο που περιέχει τις δοσμένες κορυφές.



Σχήμα 6: Ένα κανονικό 3-simplex ή Τετράεδρο.

Τα simplices γενικά δεν χρησιμοποιούνται σε αυτήν την μέθοδο όμως μια ερμηνεία είναι στο ότι δρα σε απλούς κώνους οι οποίοι μετατρέπονται σε κανονικά simpleces με έναν επιπλέον περιορισμό. Το σχήμα του πολύτοπου καθορίζεται απο τους περιορισμούς που τίθενται πάνω στην αντικειμενική συνάρτηση.

Η μορφή των προβλημάτων που δρα ο αλγόριθμος simplex είναι η κανονική γραμμική μορφή του προβλήματος δηλαδή:

$$minimize \quad c^T x \qquad\qquad subject\ to \quad Ax = b, \quad x_i \geq 0 \ ,$$

με $x = (x_1, .., x_n)$ τις μεταβλητές του προβλήματος, $c = (c_1, .., c_n)$ τους συντελεστές της αντικειμενικής συνάρτησης. Αν είχαμε ανισότητα, υπάρχει τρόπος να την μετατρέψουμε σε ισότητα με την χρήση επιπλέον μεταβλητών (σκιωδών μεταβλητών) ώστε να μην χάνεται η γενικότητα στην παρουσίαση του αλγορίθμου.

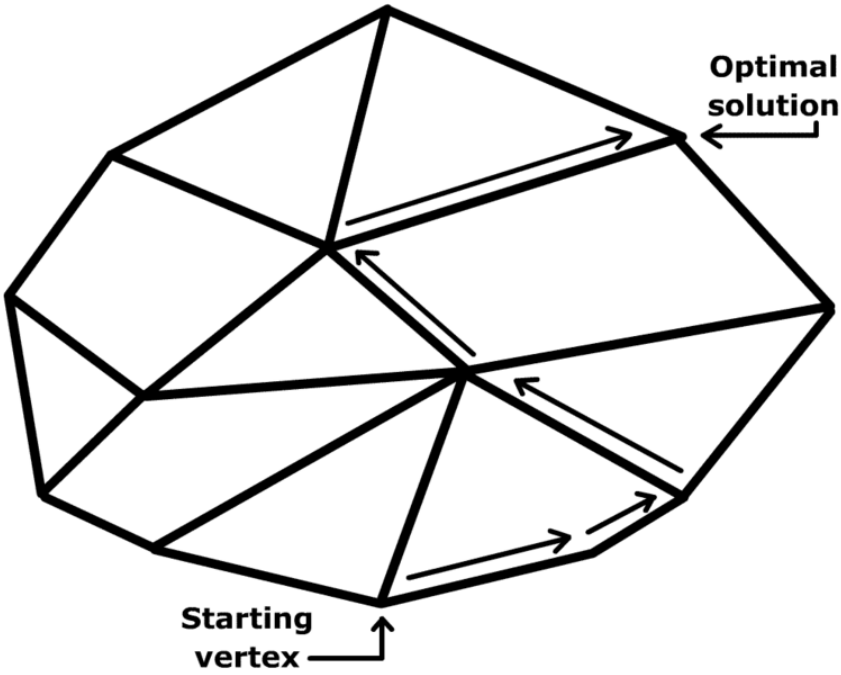
Σε γεωμετρικούς όρους, η δυνατή (εφικτή) περιοχή που καθορίζεται από την εξίσωση

$$Ax = b, x_i \geq 0$$

(3.2.1)

είναι ένα (πιθανώς μη φραγμένο) κυρτό πολύτοπο. Υπάρχει ένας απλός χαρακτηρισμός των ακραίων σημείων ή κορυφών αυτού του πολύτοπου. Συγκεκριμένα, ένα σημείο $\bar{x} = (x_1, .., x_n)$ είναι ακραίο σημείο αν και μόνο αν τα διανύσματα στήλες A_i με $x_i \neq 0$, είναι γραμμικώς ανεξάρτητα και τότε το σημείο ονομάζεται βασική εφικτή λύση (BFS).

Μπορεί να δειχθεί ότι για ένα γραμμικό πρόγραμμα σε κανονική μορφή, αν η αντικειμενική συνάρτηση έχει ελάχιστη τιμή στην εφικτή περιοχή, τότε θα έχει αυτή την τιμή σε τουλάχιστον ένα από τα ακραία σημεία. Το γεγονός αυτό μειώνει την λύση σε πεπερασμένο αριθμό υπολογισμών από την στιγμή που έχει πεπερασμένο αριθμό ακραίων σημείων. Το πρόβλημα είναι ότι ο αριθμός των σημείων είναι τεράστιος ακόμη και για πολύ μικρά γραμμικά προγράμματα.



Σχήμα 7: Ένα σύστημα γραμμικών ανισοτήτων καθορίζει ένα πολύτοπο σαν την εφικτή περιοχή. Ο αλγόριθμος simplex ξεκινά από κορυφή και κινείται κατά μήκος των πλευρών του πολυτόπου μέχρι να φτάσει την κορυφή που δίνει την βέλτιστη λύση

Μπορεί επίσης να δειχθεί ότι αν ένα ακραίο σημείο δεν ελαχιστοποιεί την αντικειμενική συνάρτηση, τότε υπάρχει μια πλευρά που περιέχει το σημείο τέτοια ώστε η αντικειμενική συνάρτηση μειώνεται κατά την κίνηση στην πλευρά μακριά από το σημείο. Αν η πλευρά είναι πεπερασμένη τότε αυτή συνδέεται σε ένα άλλο ακραίο σημείο όπου η αντικειμενική συνάρτηση έχει μικρότερη τιμή, διαφορετικά η αντικειμενική συνάρτηση είναι μη-κάτω φραγμένη στην πλευρά, και το γραμμικό πρόγραμμα δεν έχει λύση.
Ο αλγόριθμος simplex εφαρμόζει αυτό το γεγονός με το να κινείται πάνω στις πλευρές του πολυτόπου προς ακραία σημεία με ολοένα και μικρότερες τιμές τις αντικειμενικής συνάρτησης. Αυτή η διαδικασία συνεχίζεται μέχρι να βρεθεί η λύση. Ο αλγόριθμος πάντοτε τερματίζει διότι ο αριθμός των κορυφών του πολυτόπου είναι πεπερασμένος.

Θα παρουσιάσουμε την μέθοδο simplex για επίλυση προβλημάτων γραμμικού προγραμματισμού με ένα παράδειγμα μεγιστοποίησης.

Έστω το πρόβλημα: $maximize\ f(x_1, x_2) = 6x_1 + 5x_2$

$$subject\ to:\ x_1 + x_2 \leq 5$$
$$3x_1 + 2x_2 \leq 12$$
$$x_1, x_2 \geq 0$$

Θα μετασχηματίσουμε το πρόβλημα στην κανονική μορφή, και θα το κάνουμε αυτό με την εισαγωγή 2 νέων μεταβλητών (ο αριθμός 2 αντιστοιχεί στον αριθμό τον ανισοϊσοτήτων που θέλουμε να μετατρέψουμε σε ισότητα) x3,x4 οι οποίες ονομάζονται και σκιώδης τιμές.
Έτσι λοιπόν έχουμε την ακόλουθη ισοδύναμη μορφή:

$$maximize\ f(\bar{x}) = [6 \quad 5 \quad 0 \quad 0][\bar{x}]$$
$$sub.to: \begin{bmatrix} 1 & 1 & 1 & 0 \\ 3 & 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 5 \\ 12 \end{bmatrix}$$
$$[\bar{x}] = (x_1, x_2, x_3, x_4) \geq 0$$

Οι x_1, x_2 θεωρούνται μη βασικές μεταβλητές ενώ οι x_3, x_4 θεωρούνται βασικές.

Παρατήρηση: Στην μέθοδο απαιτείται να μην έχουμε αρνητικές τιμές στο δεξί μέλος των ανισοτήτων. Εάν συμβαίνει αυτό τότε πρέπει να αντιστρέψουμε την ανισότητα πολλαπλασιάζοντας με (-1).

1η επανάληψη: λύνουμε ώς προς x_3, x_4 και έχουμε

$x_3: 5 - x_1 - x_2$

$x_4: 12 - 3x_1 - 3x_2$

f: $6_{x1} + 5_{x2}$

Διαλέγουμε να αυξήσουμε την x_1 γιατί έχει τον μεγαλύτερο συντελεστή και τώρα αυτή θεωρείται καινούργια βασική μεταβλητή. Όμως δεν μπορούμε να την αυξήσουμε όσο θέλουμε διότι διαφορετικά οι σκιώδεις τιμές θα γίνουν αρνητικές, που σημαίνει παραβίαση των περιορισμών.

$2^{η}$ επανάληψη:

$$3x_1 = 12 - 2x_2 - x_4$$

$$x_1 = 4 - \frac{2}{3}x_2 - \frac{1}{3}x_4$$

$$x_3 = 5 - \left(4 - \frac{2}{3}x_2 - \frac{1}{3}x_4\right) - x_2 = 1 - \frac{1}{3}x_2 + \frac{1}{3}x_4$$

άρα

$$f = 6\left(4 - \frac{2}{3x_2} - \frac{1}{3}x_4\right) + 5x_2 = 24 + x_2 - 2x_4$$

τώρα παρατηρούμε ότι η f βελτιώνεται μόνο με την αύξηση της x_2 .

Επομένως πρέπει να βρούμε μέχρι πόσο μπορούμε να αυξήσουμε την τιμή της.

Έτσι έχουμε από τον παραπάνω πρώτο περιορισμό $x_1 \geq 0 \leftrightarrow x_2 \in ([0,6]$

ενώ από τον δεύτερο περιορισμό έχουμε $x_3 \geq 0 \leftrightarrow x_2 \in [0,3]$

επομένως επιλέγουμε $x_2 = 3$.

3η επανάληψη:

$$\frac{1}{3}x_2 = 1 - x_3 + \frac{1}{3}x_4$$

$$x_2 = 3 - 3x_3 + x_4$$

$$x_1 = 4 - \frac{2}{3}(3 - 3x_3 + x_4) - \frac{1}{3}x_4 = 2 + 2x_3 - x_4$$

άρα

$$f = 24 + (3 - 3x_3 + x_4) - 2x_4 = 27 - 3x_3 - x_4$$

και εδώ οι x_1, x_2 είναι οι βασικές μεταβλητές επομένως έχουμε την τελική λύση του προβλήματος:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ f \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 0 \\ 0 \\ 27 \end{bmatrix}$$

Απόδοση του αλγορίθμου Simplex:

Η μέθοδος simplex έχει αποδειχθεί αξιοσημείωτα αποδοτική στην πράξη και αποτέλεσε μεγάλη βελτίωση στην λύση τέτοιων προβλημάτων σε σχέση με παλιότερες μεθόδους όπως η απαλοιφή Fourier-Motzkin. Μολαταύτα, το 1972 οι Klee και Minty έδειξαν οτι η χειρότερη δυνατή πολυπλοκότητα της μεθόδου του Dantzig είναι εκθετικού χρόνου. Από τότε, για σχεδόν οποιαδήποτε παραλλαγή της μεθόδου έχει δειχθεί ότι υπάρχει μια οικογένεια γραμμικών προβλημάτων όπου δεν θα αποδίδει καλά. Είναι ακόμη ανοιχτό το ερώτημα εάν υπάρχει παραλλαγή πολυωνυμικού χρόνου η ακόμη και υπο-εκθετικού χρόνου σαν χειρότερο σενάριο πολυπλοκότητας.

Η ανάλυση και η ποιοτικοποίηση της παρατήρησης οτι ο αλγόριθμος simplex είναι αποδοτικός στην πράξη παρά την μεγάλη περιπλοκή του, οδήγησε στην ανάπτυξη άλλων μέτρων πολυπλοκότητας. Ο αλγόριθμος simplex έχει πολυωνυμικού χρόνου μέσης περίπτωσης περιπλοκότητα για διάφορες συναρτήσεις κατανομής πιθανότητας, ενώ η ακριβής απόδοσή του για την μέση περίπτωση εξαρτάται από την επιλογή της συνάρτησης πιθανότητας για τους τυχαίους πίνακες. Μια άλλη προσέγγιση χρησιμοποιεί την θεωρία κατηγορίας του Baire από την γενική τοπολογία και δείχνει οτι τοπολογικά οι περισσότεροι πίνακες μπορούν να επιλυθούν από τον αλγόριθμο simplex σε πολυωνυμικό αριθμό βημάτων.

3.2.3. Η μέθοδος των ελλειψοειδών:

Ο Khachiyan (1979) ήταν εκείνος ο οποίος, με τη μέθοδο των ελλειψοειδών που ανέπτυξε, απέδειξε ότι τα προβλήματα του γραμμικού προγραμματισμού είναι δυνατόν να λυθούν σε πολυωνυμικό χρόνο. Η μέθοδος αυτή, σε αντίθεση με τον αλγόριθμο Simplex που κινείται από κορυφή σε κορυφή, προσεγγίζει τη βέλτιστη λύση μέσα από μια ακολουθία συρρικνούμενων ελλειψοειδών.

Η μέθοδος των ελλειψοειδών έτυχε της έντονης προσοχής των ερευνητών και όχι μόνο. Είναι χαρακτηριστικό ότι οι New York Times, όπως και άλλα έντυπα γενικότερου ενδιαφέροντος δημοσίευσαν άρθρα σχετικά με την ανακάλυψη του Khachiyan, ένδειξη της πρακτικής σπουδαιότητας του γραμμικού προγραμματισμού.Κατά παράδοξο τρόπο, η μέθοδος του Khachiyan αποδείχθηκε στην πράξη χειρότερη από τη μέθοδο Simplex, η οποία λύνει ένα πρόβλημα γραμμικού προγραμματισμού σε χρόνο που είναι κατά μέσο όρο πολυωνυμικός. Ο Strang (1995) ισχυρίζεται ότι, για κάποιο λόγο που βρίσκεται κρυμμένος πίσω από τη γεωμετρία των πολυδιάστατων πολυέδρων, τα εφικτά σύνολα τύπου Klee-Minty είναι σπάνια και η μέθοδος Simplex αποδεικνύεται στην πράξη πολύ τυχερή.
Συνοπτικά, ο αλγόριθμος παρουσιάζεται στα παρακάτω βήματα:

Ο ΑΛΓΟΡΙΘΜΟΣ ΤΩΝ ΕΛΛΕΙΨΟΕΙΔΩΝ	
Βήμα 1 **Εκκίνηση του αλγορίθμου** **Βήμα 2** **Έλεγχος σύγκλισης**	► Θέσε $\bar{x}_0=0$, $Q_0 = 2^L I$, $k=0$ ► Αν $A\bar{x}_k<b$ τερμάτισε: μια εφικτή λύση έχει βρεθεί. Διαφορετικά πήγαινε στο Βήμα 3.
Βήμα 3 **Έλεγχος ασυμβίβαστου των γραμμικών ανισώσεων**	► Αν $k > 6L(n+1)^2$, τερμάτισε: το εφικτό σύνολο είναι κενό. Διαφορετικά πήγαινε στο Βήμα 4.
Βήμα 4 **Προσδιορισμός νέου ελλειψοειδούς**	► Διάλεξε οποιαδήποτε ανισότητα για την οποία είναι $a_i^t \bar{x}_k < b_i$, $i \in \{1,\ldots, m\}$ ► Θέσε $\bar{x}_{k+1} = \bar{x}_k - \frac{1}{n+1}\frac{Q_k a_i}{\sqrt{a_i^t Q_k a_i}}$ $Q_{k+1} = \frac{n^2}{n^2-1}\left(Q_k - \frac{2}{n+1}\frac{Q_k a_i a_i^t Q_k}{a_i^t Q_k a_i}\right)$
Βήμα 5 **Επανάληψη της διαδικασίας**	► Θέσε $k \leftarrow k+1$ ► Επίστρεψε στο Βήμα 2

3.2.4. Μέθοδοι Εσωτερικού Σημείου (interior point methods):

Στο κεφάλαιο αυτό παρουσιάζονται οι μέθοδοι εσωτερικού σημείου, οι οποίες στηρίζονται σε μια εντελώς διαφορετική αντίληψη από εκείνη της μεθόδου Simplex. Οι μέθοδοι εσωτερικού σημείου πρωτοεμφανίστηκαν στα μέσα της δεκαετίας του 1980 και έκτοτε έχουν αποτελέσει αντικείμενο έντονης και γόνιμης ερευνητικής δραστηριότητας. Οι μέθοδοι εσωτερικού σημείου αποτελούν την πιο εντυπωσιακή εξέλιξη στην πρόσφατη ιστορία του γραμμικού προγραμματισμού. Σε αντίθεση με τη μέθοδο Simplex, η οποία μετακινείται στο σύνορο και, πιο συγκεκριμένα, κατά μήκος των ακμών του εφικτού συνόλου, οι μέθοδοι εσωτερικού σημείου παράγουν ακολουθίες σημείων τα οποία προσεγγίζουν τη βέλτιστη λύση από το εσωτερικό του εφικτού συνόλου.

Η ανακοίνωση από τον Karmakar (1984) ενός αλγορίθμου με πολυωνυμικό χρόνο σύγκλισης, του οποίου οι επαναλήψεις διατηρούσαν τα παραγόμενα σημεία στο εσωτερικό του εφικτού συνόλου, άνοιξε νέους και συναρπαστικούς δρόμους έρευνας τόσο στην περιοχή της υπολογιστικής πολυπλοκότητας όσο και στην περιοχή του μαθηματικού προγραμματισμού. Έκτοτε σημειώνεται έντονο ερευνητικό ενδιαφέρον για μεθόδους των οποίων οι επαναλήψεις παράγουν σημεία που να ικανοποιούν ως γνήσιες ανισότητες τουλάχιστον τους περιορισμούς μη αρνητικότητας του προβλήματος. Η ερευνητική αυτή προσπάθεια οδήγησε στην ανάπτυξη αλγορίθμων οι οποίοι αποδείχθηκαν ανταγωνιστικοί με τη μέθοδο Simplex, τόσο στη θεωρία όσο και στην πράξη. Ειδικότερα μάλιστα για προβλήματα μεγάλης διάστασης, οι αλγόριθμοι εσωτερικού σημείου φαίνεται να υπερέχουν ακόμα και των καλύτερων παραλλαγών της μεθόδου Simplex. Έτσι, αν και αγορά εξακολουθεί ακόμα να κυριαρχείται από κώδικες βασισμένους στη μέθοδο Simplex, έχουν ήδη εμφανιστεί προϊόντα λογισμικού εσωτερικού σημείου, όπως είναι οι κώδικες OB1 και OSL.
Ο πρωτεύων δυικός αλγόριθμος εσωτερικού σημείου:

Έστω το πρόβλημα:

$$maximize\ z = c^T x$$
$$subject\ to\ Ax = b, x \geq 0$$

τότε το δυικό του είναι το:

$$minimize\ y = b^T w$$
$$subject\ to\ A^T w \geq c$$

Μετά την αναγωγή του δυϊκού σε τυπική μορφή με τη βοήθεια των μεταβλητών περιθωρίου s, το πρόβλημα αυτό γράφεται:

$$minimize\ y = b^T w$$
$$subject\ to\ A^T w - s = c, s \geq 0$$

Έστω τώρα xk, μια εφικτή λύση του πρωτεύοντος προβλήματος (Π) και (wk, sk) μια εφικτή λύση του δυϊκού του (Δ). Οι λύσεις αυτές είναι βέλτιστες εφόσον ικανοποιούν τις συμπληρωματικές συνθήκες χαλαρότητας:

$$(A^T w_k)_j > c_j \rightarrow (x_k)_j = 0\ (j = 1, .., n)$$

ή ισοδύναμα

$$(x_k)_j(s_k)_j = 0, (j = 1, .. n)$$

Η κεντρική ιδέα πίσω από τον πρωτεύοντα-δυϊκό αλγόριθμο εσωτερικού σημείου είναι η παραγωγή μιας ακολουθίας $\{x_k, (w_k, s_k)\}$ ζευγών λύσεων οι οποίες να κινούνται στο εσωτερικό του εφικτού συνόλου του κάθε προβλήματος και να ικανοποιούν σε ολοένα και μεγαλύτερο βαθμό τις συμπληρωματικές συνθήκες χαλαρότητας. Ειδικότερα, κάθε επανάληψη του αλγορίθμου κατασκευάζει διανύσματα $x_k(\theta_K), w_k(\theta_k)$ και $s_k(\theta_\kappa)$ τα οποία, για κάποια θετική τιμή της παραμέτρου θ_K, ικανοποιούν τις σχέσεις:

$$Ax_k = b, x_k \geq 0$$

$$A^T w_k - s_k = c, s_k \geq 0$$

$$(x_k)_j(s_k)_j = \theta_k \ (j = 1, .., n)$$

Η τιμή της παραμέτρου θ_K, από επανάληψη σε επανάληψη ελαττώνεται, μέχρις ότου επιτευχθεί ικανοποιητικός βαθμός σύγκλισης. Είναι φανερό ότι για θk > 0, η συνθήκη $x_{k_j} s_{k_j} = \theta_K$, εγγυάται ότι $x_{k_j} > 0$ και $s_{k_j} > 0$ (j = 1,..., n),

δηλαδή ότι τα σημεία $x_k(\theta_K), w_k(\theta_k))$ παραμένουν στο εσωτερικό του εφικτού συνόλου του κάθε προβλήματος. Σημειώνουμε ότι εισάγοντας τους διαγώνιους πίνακες:

$$X_k = \begin{bmatrix} (x_k)_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & (x_k)_n \end{bmatrix}, \qquad S_k = \begin{bmatrix} (s_k)_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & (s_k)_n \end{bmatrix}$$

μπορούμε να θέσουμε τις συμπληρωματικές συνθήκες χαλαρότητας στη διανυσματική μορφή

$$X_k S_k e = \theta_\kappa e$$

Έστω τώρα x_k > 0, wk και sk > 0 οι τρέχουσες εκτιμήσεις των βέλτιστων λύσεων. Οι λύσεις αυτές, ως εφικτές, ικανοποιούν τις περιοριστικές εξισώσεις

$$Ax_k = b \ \kappa\alpha\iota \ A^T w_k - s_k = c$$

των δύο προβλημάτων. Ο πρωτεύων-δυϊκός αλγόριθμος εσωτερικού σημείου βρίσκει μετατοπίσεις Δx_k, Δw_k και Δs_k τέτοιες ώστε τα διανύσματα $x_k + \Delta x_k$, $w_k + \Delta w_k$ και $s_k + \Delta s_k$ να ικανοποιούν με

μεγαλύτερη ακρίβεια τις συμπληρωματικές συνθήκες χαλαρότητας.
Εδώ ας παρατηρήσουμε ότι τα διανύσματα xk + Δxk, wk + Δwk και sk + Δsk πρέπει κατ' αρχάς να είναι εφικτά για τα δύο προβλήματα. Πρέπει δηλαδή να ικανοποιούν τις σχέσεις:

$Axk+\Delta xk=b\rightarrow Axk+A\Delta xk=b\rightarrow A\Delta xk=0$

$$A^T(w_k+\Delta w_k)-(s_k+\Delta s_k)=c\rightarrow A^Tw_k-s_k+A^T\Delta w_k-\Delta s_k=c\rightarrow$$
$$A^T\Delta w_k=\Delta s_k=0$$

Τα οποία είναι δύο συστήματα γραμμικά ως προς Δxk, Δwk και Δsk.
Τα διανύσματα xk + Δxk και sk + Δsk πρέπει επιπλέον να ικανοποιούν με μεγαλύτερη ακρίβεια και τις συμπληρωματικές συνθήκες χαλαρότητας, δηλαδή τις σχέσεις:

$$\left[x_{k_j}+\Delta x_{k_j}\right]*\left[s_{k_j}+\Delta s_{k_j}\right]=\theta_{k+1}\rightarrow$$
$$s_{k_j}\Delta x_{k_j}+x_{k_j}\Delta s_{k_j}+\Delta x_{k_j}\Delta_{s_{k_j}}=\theta_{\kappa+1}-x_{k_j}s_{k_j},\qquad j=(1,..n)$$

όπου $\theta_{\kappa+1}<\theta_\kappa$ Από τις πιο πάνω σχέσεις μπορούμε να παραλείψουμε τον όρο (Δxk)j (Δsk)j με το σκεπτικό ότι οι μετατοπίσεις Δxk και Δsk είναι πολύ μικρές και συνεπώς ο όρος (Δxk)j (Δsk)j θα πρέπει να είναι ακόμα πιο μικρός, σε κάθε περίπτωση μάλιστα μικρότερος συγκριτικά με τους όρους $s_{k_j}\Delta x_{k_j}$ και $x_{k_j}\Delta s_{k_j}$. Έτσι έχουμε

$$s_{k_j}\Delta x_{k_j}+x_{k_j}\Delta s_{k_j}=\theta_{\kappa+1}-x_{k_j}s_{k_j},\qquad j=(1,..n)$$

και σε διανυσματική μορφή:

$$S_k\Delta x_k+X_k\Delta s_k=\theta_{\kappa+1}e-X_kS_ke$$

Επομένως ο προσδιορισμός μεταπτώσεων $\Delta x_k, \Delta w_k$ και Δs_kτέτοιων ώστε τα σημεία $x_k+\Delta x_k, w_k+\Delta w_k, s_k+\Delta s_k$ να ικανοποιούν τις πρωτεύουσες δυικές συνθήκες (ΠΔ) ανάγεται στην επίλυση γραμμικών συστημάτων

$$A\Delta x_k=0$$
$$A^T\Delta w_k-\Delta s_k=0$$
$$S_k\Delta x_k+X_k\Delta s_k=\theta_{\kappa+1}e-X_kS_ke$$

Απο την δεύτερη απο τις παραπάνω σχέσεις, προκύπτει ότι $\Delta s_k=A^T\Delta w_k$

Αντικαθιστώντας την έκφραση αυτή για το Δs_k στην τρίτη των παραπάνω σχέσεων λαμβάνουμε:

$$S_k \Delta x_k + X_k A^T \Delta w_k = \theta_{\kappa+1} e - X_k S_k e$$

Πολλαπλασιάζοντας την τελευταία σχέση απο τα αριστερά με τον πίνακα AS_k^{-1} και αναγνωρίζοντας ότι $A\Delta x_k = 0$ έχουμε

$$A\Delta x_k + AS_k^{-1} X_k A^T \Delta w_k = AS_k^{-1}(\theta_{k+1} e - X_k S_k e) \rightarrow$$
$$\rightarrow AS_k^{-1} X_k A^T \Delta w_k = AS_k^{-1}(\theta_{k+1} e - X_k S_k e)$$

Ορίζοντας τώρα το διαγώνιο πίνακα D_k και το διάνυσμα $v_k(\theta_{k+1}$
όπου $D_k = S_k^{-1} X_k$, $v_k(\theta_{k+1}) = \theta_{\kappa+1} e - X_k S_k e$

προκύπτει:

$$(AD_k A^T)\Delta w_k = AS_k^{-1}\, v_k(\theta_{k+1}) \rightarrow \Delta w_k = (AD_k A^T)^{-1} A\, S_k^{-1} v_k(\theta_{k+1})$$
$$\Delta s_k = A^T \Delta w_k \rightarrow \Delta s_k = A^T (AD_K A^T)^{-1} A\, S_k^{-1} v_k(\theta_{k+1})$$
$$S_k \Delta x_k + X_k \Delta s_k = v_k(\theta_{k+1}) \rightarrow \Delta x_k = S_k^{-1} v_k(\theta_{k+1}) - S_k^{-1} X_k \Delta s_k \rightarrow$$
$$\Delta x_k = S^{-1} v_k(\theta_{k+1}) - D_k \Delta s_k$$

Όπως αποδεικνύεται, αν η παράμετρος θ από επανάληψη σε επανάληψη ενημερώνεται σύμφωνα με τη σχέση $\theta_{\kappa+1} = \alpha\, \theta_\kappa$,0 < α < 1 τότε κάτω από ορισμένες προϋποθέσεις για τον πολλαπλασιαστικό συντελεστή α ο αλγόριθμος συγκλίνει σε χρόνο πολυωνυμικό. Οι Monteiro και Adler (1989), στους οποίους οφείλεται ο αλγόριθμος που περιγράψαμε στην ενότητα αυτή, απέδειξαν ότι θέτοντας:

$$\alpha = 1 - \frac{3.5}{\sqrt{n}}$$

ο αλγόριθμος βρίσκει τη βέλτιστη λύση μετά από O($\sqrt{n}$ L) το πολύ επαναλήψεις. Το εύρημα αυτό αποτελεί ένα πολύ καλό θεωρητικό φράγμα. Στην πράξη όμως η συγκεκριμένη τιμή του α δεν επιτρέπει τη γρήγορη μείωση της παραμέτρου θ και, έτσι, ο αλγόριθμος χρειάζεται πολλές επαναλήψεις μέχρις ότου φθάσει σε μια τόσο μικρή τιμή για το θ ώστε η διαφορά της τιμής της αντικειμενικής συνάρτησης του πρωτεύοντος από την τιμή της αντικειμενικής

συνάρτησης του δυϊκού του να είναι πολύ κοντά στο μηδέν. Για να πάρουμε έναν πρακτικά πιο αποτελεσματικό αλγόριθμο, θα έπρεπε να μειώνουμε ταχύτερα την τιμή της παραμέτρου θ. Μια μεγάλη όμως μείωση της τιμής του θ θα είχε ως αποτέλεσμα οι νέες εκτιμήσεις των βέλτιστων λύσεων να μην είναι γνήσια θετικές.

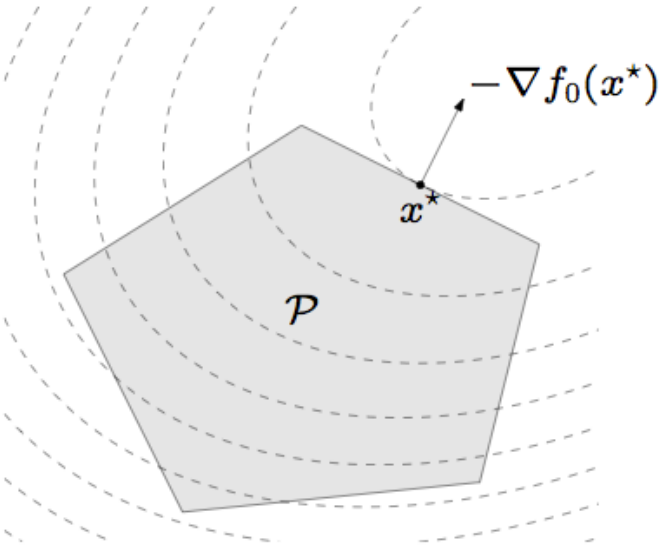
4. Τετραγωνικός Προγραμματισμός:

4.1. Εισαγωγή

Δομή: $minimize \quad \frac{1}{2}x^T Px + q^T x + r$

$subject\ to \quad Gx \leq h$

με $P \in S_+^n$ ενώ το σύνολο εφικτών λύσεων είναι το:



Σχήμα 8: Ισοσταθμικές και πεδίο ορισμού όπου οι ισοσταθμικές είναι ελλειψοειδή

Ένα παράδειγμα αναλυτικής επίλυσης τέτοιου προβλήματος είναι η περίπτωση του γραμμικού τετραγωνικού ρυθμιστή (LQR) όπου η αντικειμενική συνάρτηση είναι της μορφής:

$$l\big(x(t), u(t)\big) = x(t)^T Q x(t) + u(t)^T R u(t)\ , \quad Q, R \geq 0 \tag{4.1}$$

και μπορεί να λυθεί μέσω δυναμικού προγραμματισμού καθώς η βέλτιστη τιμή δίνεται απο την σχέση: $V(x_0) = x_0^T P x_0$

όπου $P = Q + A^T PA - A^T PB(R + B^T PB)^{-1} B^T PA$ (εξίσωση Riccati)

και τελικά η βέλτιστη ρύθμιση είναι μια γραμμική ανατροφοδότηση των μεταβλητών κατάστασης όπου $u^*(t) = -(R + B^T PB)^{-1} B^T PAx(t)$

4.2. Η μέθοδος Newton-συνδυασμός με interior-point μέθοδο

Έστω το πρόβλημα: $minimize\ f(x)$

$subject\ to\ Ax = b$

Τότε το βήμα της f για εφικτό x δίνεται απο την λύση του συστήματος:

$$\begin{bmatrix} \nabla^2 f(x) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ w \end{bmatrix} = \begin{bmatrix} -\Delta f(x) \\ 0 \end{bmatrix} \tag{4.2.1}$$

και παρατηρούμε ότι κάθε Newton βήμα έχει μεγάλο υπολογιστικό κόστος για μεγάλα προβλήματα καθώς περιλαμβάνει τον υπολογισμό σε κάθε επανάληψη της Hessian μήτρας. Αυτή η μέθοδος έχει το πλεονέκτημα ότι σε κάθε επανάληψη η συνάρτηση θα έχει μικρότερη τιμή απο ότι είχε στην προηγούμενη. καθώς και οτι γενικά δεν απαιτεί μεγάλο αριθμό επαναλήψεων.

Στην περίπτωση όπου έχουμε και περιορισμούς της μορφής $f_i(x) \leq 0$
και ισχύουν οι συνθήκες του Slater, μπορούμε να αντιμετωπίσουμε το πρόβλημα με χρήση της λογαριθμικής μεθόδου φράγματος όπου μετατρέπουμε το πρόβλημα σε μορφή που θα μπορούμε να εφαρμόσουμε την μέθοδο Newton δηλαδή το πρόβλημα προσεγγίζεται από το:

$$minimize\ f_0(x) - \frac{1}{t}\sum_{i=1}^{m} \log\left(-f_i(x)\right)$$

$$subject\ to\ Ax = b \tag{4.2.2}$$

και ταυτίζεται με το αρχικό πρόβλημα όταν $t \to \infty$
επομένως η επιλογή της τιμής του t βασίζεται σε ευριστικές τεχνικές

4.3 Active-Set αλγόριθμοι

Οι μέθοδοι ενεργού συνόλου, είναι επαναληπτικές μέθοδοι που επιλύουν μια ακολουθία ισοτικών περιορισμών τετραγωνικών υποπροβλημάτων. Στόχος της μεθόδου είναι η πρόβλεψη του ενεργού συνόλου των περιορισμών που ικανοποιούνται με ισότητα στην λύση του προβλήματος. Η συνηθισμένη μέθοδος ενεργού συνόλου χωρίζεται σε 2 φάσεις. Η πρώτη εστιάζει στην εφικτότητα ενώ η δεύτερη εστιάζει στην βελτιστοποίηση. Ένα πλεονέκτημα αυτών των αλγορίθμων είναι ότι είναι ιδανικοί για “warm starts” όπου θέτουμε μία καλή εκτίμηση για το βέλτιστο ενεργό σύνολο στην εκκίνηση του αλγορίθμου. Αυτό είναι ιδιαίτερα χρήσιμο όταν πρέπει να λύσουμε μία ακολουθία προβλημάτων τετραγωνικού προγραμματισμού όπως για παράδειγμα σε εφαρμογές MPC.

Γενικά η δομή ενός active set αλγορίθμου είναι η ακόλουθη:

Όσο «όχι ικανοποιητικά βέλτιστο»

- Λύσε προσεγγιστικά το πρόβλημα μόνο με τους ισοτικούς περιορισμούς απο το ενεργό σύνολο
- Υπολόγισε τους πολλαπλασιαστές Lagrange του ενεργού συνόλου
- Αφαίρεσε ένα υποσύνολο των περιορισμών με αρνητικούς Lagrange πολλαπλασιαστές
- Έλεγχος για μη ικανοποίηση περιορισμών

Τέλος

Σημείωση: ο αριθμός των επαναλήψεων σε αυτού του τύπου μεθόδων επίλυσης είναι μεγαλύτερος απο τις επαναλήψεις σε αλγορίθμους εσωτερικού σημείου. Παρόλα αυτά, το υπολογιστικό κόστος είναι πολύ μικρότερο σε κάθε επανάληψη και επίσης οι αλγόριθμοι εσωτερικού σημείου δεν ευνοούνται απο “warm starts”.

4.4. Γρήγορος Προβλεπτικός Έλεγχος-Fast MPC

Υπάρχουν διάφοροι τρόποι που θα μας βοηθήσουν να λύσουμε γρήγορα το πρόβλημα βελτιστοποίησης. Ένας τέτοιος τρόπος είναι με το να το λύσουμε προσεγγιστικά καθώς και να χρησιμοποιήσουμε warm start τεχνικές σε συνδιασμό με μεθόδους εσωτερικού σημείου, κάτι που θα μειώσει σε πολύ μεγάλο βαθμό τον αριθμό των επαναλήψεων. Παρακάτω παραθέτουμε έναν πίνακα απο διάφορα πειράματα που έγιναν σε προβλήματα MPC όπου γίνεται σύγκριση Fast MPC μεθόδων με συμβατικές μεθόδους. Ο πίνακας αυτός ήταν μέρος της παρουσίασης μίας εκ των διαλέξεων για το μάθημα κυρτής βελτιστοποίησης 2 που γίνεται στο Stanford στο τμήμα ηλεκτρολόγων μηχανικών απο τον καθηγητή Stephen Boyd.

problem size			QP size		run time (ms)	
n	m	T	vars	constr	fast mpc	SDPT3
4	2	10	50	160	0.3	150
10	3	30	360	1080	4.0	1400
16	4	30	570	1680	7.7	2600
30	8	30	1110	3180	23.4	3400

4.5. Μέθοδος Επιταχυνόμενης Δυϊκής Προβολής Κλίσης (GPAD):

Ο αλγόριθμος GPAD αποτελεί έναν γρήγορο τρόπο επίλυσης τετραγωνικών αυστηρά κυρτών προβλημάτων . Θα συνεχίσουμε την ανάλυση με αντικατάσταση του συμβολισμού όπως γίνεται στο [Bemporad, Patrinos 2012]

Έστω το πρόβλημα βελτιστοποίησης:

$$\begin{aligned} &minimize \quad V(z) = \frac{1}{2}z'Mz + (Cp + g)'z + \frac{1}{2}p'Yp \\ &subject\ to \quad Gz \leq Ep + b \end{aligned} \tag{4.5.1}$$

όπου p είναι οι αρχικές συνθήκες και z οι μεταβλητές απόφασης, $M \in \mathbb{S}++n$ $Hessian\ μήτρα$, $C \in \mathbb{R}^{\#states \times n}$,$g \in \mathbb{R}^m$ ενώ ο Υ δεν επηρεάζει την λύση.

Ορισμός 1:

Στόχος είναι η εύρεση λύσης $z \in \mathbb{R}^n$ τέτοια ώστε να ισχύουν οι σχέσεις:

$$V(z) - V^* \leq \varepsilon_v \quad (1_a)$$
$$\|g(z)_+\|_\infty \leq \varepsilon_g \qquad (1_b)$$

(4.5.2)

όπου

- V→η αντικειμενική συνάρτηση
- V^* η βέλτιστη λύση
- $\varepsilon_v, \varepsilon_g \geq 0$
- $g(z) = \nabla_z dual\ function$

Η εξίσωση (1_a) φράσει την απόσταση της τωρινής τιμής της αντικειμενικής συνάρτησης απο την βέλτιστη τιμή ενώ η 1_b φράσει την μέγιστη πρωτεύουσα μη-εφικτότητα για την πρωτεύουσα υπο-βέλτιστη λύση.

Ορίζουμε το δυικό πρόβλημα ώς:

$$\Psi^*(p) = \inf_y \frac{1}{2} y^T H y + (Dp + d)^T y + d_p \ \ , s.t \ \ y \geq 0$$

(4.5.3)

όπου

$H = GM^{-1}G^T, D = GM^{-1}C + E, \ d = GM^{-1}g + b\ , d_p = \frac{1}{2}(Cp + g)^T M^{-1}(Cp + g) + p^T Y p.$

Αφού το πρόβλημα είναι τετραγωνικό και κυρτό, η ισχυρή δυϊκότητα πάντοτε θα ισχύει επομένως θα έχουμε

$$V^*(p) = -\Psi^*(p).$$

(4.5.4)

Σημείωση: ενώ στις περισσότερες δυϊκές μεθόδους ο στόχος είναι να βρεθεί μια προσεγγιστική δυική λύση

$$\Psi^* - \Psi(p, y) \leq \varepsilon_\Psi$$

Σε αυτήν τη μέθοδο ο στόχος είναι να βρεθεί προσεγγιστική πρωτεύουσα λύση όπως ορίζεται με τον Ορισμό 1. Αυτό είναι πολύ σημαντικό σε εφαρμογές MPC

όπου ο στόχος είναι ο υπολογισμός της βέλτιστης ακολουθίας μεταβλητών εισόδου για το πρωτεύων πρόβλημα.

Σχόλιο 1: Η σύγκλιση του αλγορίθμου μπορεί να δειχθεί για οποιοδήποτε βήμα που ικανοποιεί την σχέση:

$$\frac{1-\theta_{v+1}}{\theta_{v+1}^2} \leq \frac{1}{\theta_v^2}$$

Αλγόριθμος GPAD σε ψευδοκώδικα:

Algorithm 1: Accelerated Dual Gradient-Projection (GPAD)

Input: $y_{(0)} = y_{(-1)} \in \mathbb{R}_+^m$. $\theta_0 = \theta_{-1} = 1$. $v \leftarrow 0$

Step 1. $w_{(v)} = y_{(v)} + \theta_v(\theta_{v-1}^{-1} - 1)(y_{(v)} - y_{(v-1)})$

Step 2. $z_{(v)} = \arg\min_{z \in \mathcal{Z}} \mathcal{L}(z, w_{(v)})$

Step 3. $y_{(v+1)} = \left[w_{(v)} + \frac{1}{L_\Psi} g(z_{(v)})\right]_+$

Step 4. $\theta_{v+1} = \frac{\sqrt{\theta_v^4 + 4\theta_v^2} - \theta_v^2}{2}$. Set $v \leftarrow v+1$ and go to Step 1.

όπου z_v είναι η μέση πρωτεύουσα ακολουθία και δίνεται από την σχέση:

$$z_v = \theta_v^{-1} \sum_{i=0}^{v} \theta_v^{-1} z_i = (1-\theta_v) z_{v-1} + \theta_v \bar{z_v}, \qquad \bar{z_v} = -M_G w_v - g_P$$

και $z_{-1} = 0$ ενώ το y δίνεται από την σχέση $y_{v+1} = [w_v + G_L \bar{z_v} + p_D]_+$,

$$y_0 = y_{-1} = 0, \quad \theta_0 = \theta_{-1} = 1$$

- v *είναι ο δείκτης της επανάληψης* $\in \mathbb{N}$
- $M_G = M^{-1}G^T$, $g_P = M^{-1}(Cp + g)$, $G_L = \frac{1}{L}G$,
- $p_D = -\frac{1}{L}(Ep + b)$, *όπου* $L = \max\big(eigenvalue(H)\big)$

Παρατήρηση: Αφότου υπολογίσουμε τους πίνακες M_G, G_L, p_D, g_P η κύρια υπολογιστική διαδικασία του GPAD είναι κατά τον υπολογισμό των STEP 2,STEP 3 που απαιτούν γινόμενο πίνακα με διάνυσμα επομένως έχουν υπολογιστικό κόστος της τάξης $O(nm)$. Για τετραγωνικά προβλήματα που προκύπτουν απο διαμόρφωση MPC αυτό αντιστοιχεί σε περιπλοκότητα $O(N^2)$

όπου N ο ορίζοντας πρόβλεψης.

Φράγμα στον αριθμό των επαναλήψεων:
Στο [Patrinos and bemporad, 2012,Theorem 2] οι συγγραφείς απέδειξαν οτι $\forall v \in \mathbb{N}_+$ ισχύει η ακόλουθη ανισότητα:

$$G\mathrm{z_v} - Ep - b \leq \frac{8L}{(v+2)^2}\|y_0 - y^*\|\begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix}$$

που δείχνει ότι το άνω φράγμα $N_g(p, \varepsilon_g)$

$$N_g(p, \varepsilon_g) = \left[\sqrt{\frac{8L}{\varepsilon_g}\|y_0 - y^*\|}\right] - 2 \tag{4.5.5}$$

στον αριθμό των επαναλήψεων εγγυάται την μέση πρωτεύουσα λύση z_v όπως έχει οριστεί απο το βήμα 2 του αλγορίθμου να είναι εφικτή μεσα σε ένα προεπιλεγμένο διάστημα $\varepsilon_g > 0$

Όσων αφορά την υποβελτιστοποίηση, η οποία είναι η απόσταση της τιμής της αντικειμενικής συνάρτησης στην επανάληψη "n' από την ελάχιστη τιμή της συνάρτησης $V^*(p)$ οι συγγραφείς της παραπάνω δημοσίευσης απέδειξαν ότι $\forall v \in \mathbb{N}_+$ ισχύει η ανισότητα

$$\frac{8L}{(v+2)^2}\|y_0 - y^*\|^2 \leq V(z_v, p) - V^*(p) \leq \frac{2L}{(v+2)^2}\|y_0 - y^*\|$$

η οποία τελικά δίνει το άνω φράγμα

$$N_g(p, \varepsilon_g) = \left[\sqrt{\frac{2L}{\varepsilon_g}\|y_0 - y^*\|}\right] - 2 \tag{4.5.6}$$

στον αριθμό των επαναλήψεων που εγγυάται την μέση πρωτεύουσα λύση z_v όπως έχει οριστεί απο το βήμα 2 του αλγορίθμου να είναι εφικτή μεσα σε ένα προεπιλεγμένο διάστημα $\varepsilon_g > 0$

Παρατήρηση: Το κάτω φράγμα στην παραπάνω ανισότητα μπορεί επίσης να είναι αξιοσημείωτο καθώς το z_v μπορεί να είναι μη εφικτό και άρα κάποιος να έχει $V(z_v) \leq V^*$.

Με βάση αυτά τα φράγματα οι συγγραφείς αναφέρουν μια δεύτερη δυνατή εκδοχή του αλγορίθμου GPAD η οποία είναι η ακόλουθη:

Algorithm 2 GPAD with fixed number of iterations

0. **init**: $y_0 = y_{-1} = 0$, $z_{-1} = 0$;
1. **for** $\nu = 0$ **to** $N_\nu(\varepsilon_V, \varepsilon_g)$
   compute w_ν, z_ν, $y_{\nu+1}$ as in (8);
2. **stop**. z_ν = primal solution, $y_{\nu+1}$ = dual solution.

Αυτός ο αλγόριθμος είναι αρκετά επαρκής για χρήση σε δύσκολα real-time συστήματα στα οποία κάποιος πρέπει ούτως ή άλλος να επιλέξει υπολογιστικό hardware τέτοιο ώστε να αντέχει τον χειρότερο δυνατό αριθμό επαναλήψεων N_v εντός του χρόνου δειγματοληψίας του MPC ρυθμιστή.
Με αυτό το σκεπτικό, ο αλγόριθμος 2 είναι προτιμότερος του 1 καθώς αποφεύγει τον υπολογισμό του δυικού κενού καθώς και το ποσό της παραβίασης των περιορισμών κάτι που κάνει κάθε επανάληψη απλούστερη αλλά και τον κώδικα περισσότερο συμπαγή.

Οι παραπάνω αλγόριθμοι έχουν την ενδιαφέρουσα ιδιότητα να επιστρέφουν την ακριβή βέλτιστη λύση σε περίπτωση που το ελάχιστο βρίσκεται εντός της κρίσιμης περιοχής CR_0 που αντιστοιχεί στο να είναι όλοι οι περιορισμοί μη ενεργοί.
δλδ:

$$CR_0 = \{p \in \mathbb{R}^{\mathbb{m}_x} : -(GM^{-1}C + E)p \leq b - GM^{-1}g\}$$

όπου σε αυτή την περίπτωση έχουμε $y^*(p) = 0$
και η ακριβής βέλτιστη λύση είναι η $z^*(p) = -M^{-1}Cp + g$ ενώ ο αλγόριθμος 1 επιστρέφει $w_v = 0, z_v == -M^{-1}Cp + g, y_{v+1} = [p_D]_+ = 0 \; \forall v \in \mathbb{N}$
άρα η βέλτιστη λύση βρίσκεται σε μία επανάληψη. Αυτή είναι μια χρήσιμη ιδιότητα καθώς οι τοπικές ,κλειστού βρόγχου ,ιδιότητες σταθερότητας του MPC

διατηρούνται. Για ενίσχυση των παγκόσμιων, κλειστού βρόγχου, ιδιοτήτων σταθερότητας, θα πρέπει να διαμορφώσουμε έναν νόμο ρύθμισης MPC ο οποίος να είναι εύρωστος σε σχέση με τις τιμές των $\varepsilon_v, \varepsilon_g$, ένα θέμα το οποίο ερευνάται αυτή την στιγμή.

5. Διαμόρφωση του προβλήματος QP σε προβλήματα προβλεπτικού ελέγχου

Στο παρόν κεφάλαιο παρουσιάζεται η διαμόρφωση του προβλήματος QP σε προβλήματα MPC. Σκοπός μας, είναι η ελαχιστοποίηση μίας συνάρτησης της μορφής

$$V(x) = \sum_{k=0}^{N-1} l(x_k, u_k) \qquad (5)$$

s.t.

$$x_{k+1} = A_k x_k + B_k u_k + f_k \qquad (5a)$$

$$F_k x_k + G_k u_k \leq c_k \qquad (5b)$$

$$F_N x_N \leq c_N \quad \text{(Terminal cost)} \qquad (5c)$$

όπου x_k είναι το διάνυσμα μεταβλητών κατάστασης (διαστάσεων $1\times n$) ενώ το u_k είναι το διάνυσμα μεταβλητών εκ-χειρισμού $(1\times m)$, την χρονική στιγμή k.
Τέλος, N είναι ο χρονικός ορίζοντας και ισχύει ότι k=1….N-1

(Ο όρος f_k μπορεί να αντιπροσωπεύει τον θόρυβο του συστήματος).

Το διάνυσμα που θέλουμε να βελτιστοποιήσουμε (έτσι ώστε η (1) να γίνεται ελάχιστη) είναι το
Z=**u**=[u'$_0$,u'$_1$,……u'$_{N-1}$]
Προφανώς ισχύει ότι $z \in \mathbb{R}^{m\times N}$ όπου m είναι ο αριθμός των μεταβλητών εκ-χειρισμού (ή αλλιώς ο αριθμός των στοιχείων που περιέχει το διάνυσμα u).

Ισχύει ότι: $l_k(x,u) = \frac{1}{2}[x' \quad u']\begin{bmatrix} Q_k & S_k' \\ S_k & R_k \end{bmatrix}\begin{bmatrix} x \\ u \end{bmatrix} + [q_k' \quad r_k']\begin{bmatrix} x \\ u \end{bmatrix} + s_k$

όπου $x', u' κλπ = x^{\mathrm{T}}, u^{\mathrm{T}}$ *κλπ δηλαδή είναι οι αντίστοιχοι ανάστροφοι πίνακες*

Κάνοντας πράξεις έχουμε:

$$l_k(x,u) = \frac{1}{2}[x'Q + u'S \quad x'S' + u'R]\begin{bmatrix} x \\ u \end{bmatrix} + (q'x + r'u) + s$$

$$= \frac{1}{2}\{(x'Q + u'S)x + (x'S' + u'R)u\} + q'x + r'u + s$$

$$= \frac{1}{2}x'Qx + \frac{1}{2}u'Sx + \frac{1}{2}x'S'u + \frac{1}{2}u'Ru + q'x + r'u + s \qquad (\mathbf{5.1})$$

Ας θεωρήσουμε αρχική κατάσταση την $\mathbf{x_0}$ όπου αυτή είναι διάνυσμα της μορφής

$xo = \begin{bmatrix} x_{01} \\ x_{02} \\ x_{03} \\ \dots \\ x_{0n} \end{bmatrix}$. Στήν επόμενη χρονική στιγμή, η κατάσταση βρίσκεται από μία γραμμική εξίσωση της μορφής : $x_1 = A_0x_0 + B_0u_0 + f_0$ **(5.2)**

όπου οι πίνακες A,B,f, προκύπτουν από τα χαρακτηριστικά του συστήματος ενώ το u_0 είναι το αρχικό διάνυσμα που περιέχει τις μεταβλητές εκ-χειρισμού.
Την επόμενη χρονική στιγμή t_2 , το διάνυσμα μεταβλητών κατάστασης δίνεται από τη σχέση : $x_2 = A_1x_1 + B_1u_1 + f_1$

Αν αντικαταστήσουμε όπου x1 στην από πάνω σχέση την σχέση (5.2) έχουμε :

$$x_2 = A_1x_1 + B_1u_1 + f_1$$
$$= A_1(A_0x_0 + B_0u_0 + f_0) + B_1u_1 + f_1$$
$$= A_1A_0x_0 + A_1B_0u_0 + A_1f_0 + B_1u_1 + f_1$$

Αντίστοιχα υπολογίζουμε την κατάσταση x_3 τη χρονική στιγμή t_3
$x_3 = A_2x_2 + B_2u_2 + f_2$

$= A_2A_1A_0x_0 + A_2A_1B_0u_0$ +$A_2B_1u_1$+B_2u_2+$A_2A_1f_0$ A_2f_1+B_2f_2+f_2

Τελικά, παρατηρούμε ότι μπορούμε να εκφράσουμε σε μορφή πινάκων τις παραπάνω εξισώσεις λαβαίνοντας υπόψη ότι ο αναδρομικός τύπος είναι της μορφής: $x_n = A_{n-1}x_{n-1} + B_{n-1}u_{n-1} + f_{n-1}$

A_bar B_bar

$$\begin{bmatrix} x0 \\ x1 \\ x2 \\ x3 \\ . \\ . \\ . \\ .xN \end{bmatrix} = \begin{bmatrix} I_{nxn} \\ A_0 \\ A_1A_0 \\ A_2A_1A_0 \\ \dots\dots\dots \\ A_{N-1}..A_1A_0 \end{bmatrix} xo + \begin{bmatrix} \mathbf{0_{nxm}} & \dots & \dots & \mathbf{0} \\ Bo & \mathbf{0} & .. & \mathbf{0} \\ A1Bo & B1 & \dots & \mathbf{0} \\ A2A1Bo & A2B1 & .. & \vdots \\ \dots. & \dots & .. & B_{N-1} \end{bmatrix} \begin{bmatrix} uo \\ u1 \\ u2 \\ .. \\ .. \\ u_{N-1} \end{bmatrix} +$$

$$\begin{bmatrix} 0 & \dots & 0 \\ I_{nxn} & 0.. & 0 \\ A1 & I_{nxn} & \dots. \\ A2A1 & A2\dots & \dots. \\ \dots. & \dots & I_{nxn} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ . \\ . \\ f_{N-1} \end{bmatrix}$$

D_bar matrix

Ή σε πιο συμπυκνωμένη μορφή

$$\overline{\boldsymbol{x}} = \bar{A}\boldsymbol{x_0} + \bar{B}\overline{\boldsymbol{u}} + \overline{D}\overline{\boldsymbol{f}} \quad \textbf{(5.3)}$$

όπου $A_{bar} = \bar{A} \in \mathbb{R}^{(N+1)n\times n}$, $B_{bar} \in \mathbb{R}^{(N+1)n\times(n*m)}$, $D_{bar} \in \mathbb{R}^{(N+1)n\times(n*n)}$

'Ετσι, την χρονική στιγμή N, το διάνυσμα μεταβλητών κατάστασης x_N
με βάση τον πιο πάνω πίνακα, θα δίνεται από την σχέση :

$$\boldsymbol{x_N} = (A_{N-1}..A_1A_0)\boldsymbol{x_0} + (A_{N-1}\dots A_2A_1B_0)\boldsymbol{u_0} + (A_{N-2}..A_1B_0)\boldsymbol{u_1} + \dots + (B_N)\boldsymbol{u_{N-1}} + (A_{N-1}..A_1)\boldsymbol{f_0} + (A_{N-2}..A_1)\boldsymbol{f_1} + .. + I_{NxN}\boldsymbol{f_N}$$

Επιστρέφουμε στην σχέση (5.1) που έχουμε καταλήξει πιο πάνω δηλαδή στην :

$$(S) = l_k(x,u) = \frac{1}{2}x'Qx + x'S'u + \frac{1}{2}u'Ru + q'x + r'u + s$$

Από την σχέση (5.3), προκύπτει ότι $\overline{x'} = x_0{}'\overline{A'} + \overline{u'}\overline{B'} + \overline{f'}\overline{D'}$ **(5.4)**

Αντικαθιστώντας την (5.2) και την (5.4) όπου x και $\bar{x}$ αντίστοιχα, στην σχέση (5.1) παίρνουμε (για πρακτικούς λόγους, όλοι οι πίνακες $\bar{A}, \bar{B}$ *κλπ θα γράφονται ώς* A, B *κλπ αντίστοιχα*):

$$(5.2), (5.4) \rightarrow (5.1) = l(x,u) = \frac{1}{2}(x_0'A' + u'B' + f'D')Q(Ax_0 + Bu + Df) +$$
$$+(x_0'A' + u'B' + f'D')S'u +$$
$$\frac{1}{2}u'Ru + q'(Ax_0 + Bu + Df) + r'u +$$

Κάθε όρος του αθροίσματος που περιέχει το x θα ισούται αντίστοιχα μέ:

$$\frac{1}{2}(x_0'A' + u'B' + f'D')Q(Ax_0 + Bu + Df)$$
$$= \frac{1}{2}x_0'A'QAx_0 + \frac{1}{2}x_0'A'QBu + \frac{1}{2}x_0'A'QDf + \frac{1}{2}u'B'QAx_0$$
$$+\frac{1}{2}u'B'QBu + \frac{1}{2}u'B'QDf + \frac{1}{2}f'D'QAx_0 + \frac{1}{2}f'D'QBu$$
$$+\frac{1}{2}f'D'QDf \quad (\boldsymbol{S.a})$$

$$(x_0'A' + u'B' + f'D')S'u = x_0'A'S'u + u'B'S'u + f'D'S'u \quad (\boldsymbol{S.b})$$

$$\frac{1}{2}u'Ru + q'(Ax_0 + Bu + Df) + r'u + s$$
$$= \frac{1}{2}u'Ru + q'Ax_0 + q'Bu + q'Df + r'u + s \quad (\boldsymbol{S.c})$$

Προφανώς $l(x,u) = (S.a) + (S.b) + (S.c)$

Τελικά, μαζεύουμε όλους τους όρους που περιέχουν : $u'Πίνακες u$

Δηλαδή τους όρους:

$$\frac{1}{2}u'B'QBu,\ \ u'B'S'u,\ \ \ \frac{1}{2}u'Ru$$

Βγάζοντας κοινό παράγοντα στους όρους, έχουμε:

$$\frac{1}{2}u'(B^TQB + B^TS + R)u = \frac{1}{2}u'\boldsymbol{M}u \tag{5.5}$$

Ομοίως πράττουμε για τους όρους που είναι της μορφής $x'Πίνακες x$

Δηλαδή τον όρο:

$$\frac{1}{2}x_0'A'QAx_0 = \frac{1}{2}x_0'\boldsymbol{Y}x_0 \tag{5.6}$$

Τέλος κάνουμε την ίδια διαδικασία για τους όρους που περιέχουν το διάνυσμα μεταβλητών εκ χειρισμού, για όλες τις χρονικές στιγμές μέχρι τον ορίζοντα N, u:

$$\frac{1}{2}x_0'A'QBu,\ \ \ \frac{1}{2}f'D'QBu\ \ \frac{1}{2}x_0'A'S'u\ \ \ ,\ \ \frac{1}{2}f'D'S'u,\ \ q'Bu\ ,\ \ \ r'u,\ \ \ \frac{1}{2}u'B'QAx_0,$$
$$\frac{1}{2}u'B'QDf,\ \ \ \ \frac{1}{2}u'SAx_0, \frac{1}{2}u'SDf,$$

‘Αρα:

$$\left(\frac{1}{2}x_0'A'QB + \frac{1}{2}x_0'A'S' + \frac{1}{2}x_0'A'Q'B + \frac{1}{2}x_0'A'S'\right)u$$
$$= \frac{1}{2}x_0'(A'QB + A'S' + A'Q'B + A'S')u =$$

$$\frac{1}{2}(B^TQA + S^TA)^{\mathrm{T}}x_0u = \frac{1}{2} * \boldsymbol{C^T}u \tag{5.7}$$

και για τους όρους που δεν περιέχουν το διάνυσμα μεταβλητών κατάστασης έχουμε :

$$(B^T QDf + S^T Df + B^T q^T + r)u = \boldsymbol{g}u \tag{5.8}$$

Τελικά εκφράσαμε την αντικειμενική μας συνάρτηση σε μορφή (τετραγωνική):

$$V^*(x_0) = \min \frac{1}{2} u' Mu + (Cx_0 + g)^T u + \frac{1}{2} x_0' Y x_0$$

s.t

$$Gu \leq Ex_0 + b \tag{5.9}$$

Είναι σημαντικό να ξαναθυμίσουμε ότι μία τετραγωνική συνάρτηση όπως η παραπάνω, είναι κυρτή αν και μόνο αν ο πίνακας M είναι θετικά ορισμένος η ημιορισμένος δηλαδή

$$M \in \mathbb{S}_{++}^{m*N \times m*N}$$

και αυτή η συνθήκη προκύπτει από θεώρημα κυρτής ανάλυσης όπου πέρνουμε την παράγωγο της συνάρτησης $\nabla V = Mu + (C_{xo} + g)\hat{}T$ και μετά την μήτρα Hessian $\nabla^2 V = M$ επομένως αρκεί M>0 ώστε το πρόβλημα να είναι κυρτό.

Στη συνέχεια διαμορφώνονται οι πίνακες των περιορισμών :

$$F_k x_0 + G_k u_0 \leq c_k \tag{5.10}$$

Όπου $F_k \in \mathbb{R}^{h \times n}, G_k \in \mathbb{R}^{h \times m}, c_k \in \mathbb{R}^{h \times 1}$ με h χαρακτηριστικό ώς το πλήθος των περιορισμών κάθε χρονική στιγμή.

Θα εργαστούμε και εδώ αντίστοιχα με πριν δηλαδή:

$F_k x_0 + G_k u_0 \leq c_k$ για t=0

Αν $x_1 = Ax_0 + Bu_0 + f_0$

Τότε $F_k x_1 + G_k u_1 \leq c_k \quad or\ F_k(Ax_0 + Bu_0 + f_0) + G_k u_1 \leq c_k\ or$

$F_k B u_0 + G_k u1 \leq\ -F_k A x_0 - F_k f_0 + c_k$ για t=1

Προχωρώντας ανάλογα και για τις υπόλοιπες χρονικές στιγμές καταλήγουμε στο εξής μοτίβο:

$$\begin{bmatrix} G_k & 0 \ \dots & 0_{n\times m} \\ F_k B & G_k \ \dots & 0 \\ F_K AB & F_K B \ .. & 0 \\ \vdots & \ddots & \vdots \\ F_k A^{N-1} B & F_k A^{N-2} B \ \dots & G_K \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-1} \end{bmatrix} \leq \begin{bmatrix} F_k \\ -F_k A \\ -F_k A^2 \\ \vdots \\ -F_k A^N \end{bmatrix} [x_0] + \begin{bmatrix} c_k \\ -F_k f + c_k \\ -F_k f + c_k - F_k A f \\ \vdots \end{bmatrix}$$

(5.11)

Επίσης στην τελευταία γραμμή του πίνακα μπορούμε να τοποθετήσουμε τα terminal boundaries που περιγράφονται από την εξίσωση (4 c) που έχει τους πίνακες F_N και c_{an} . Το να συμπεριλάβουμε αυτόν τον τελευταίο περιορισμό θα είχε το ‘’αντίκτυπο’’ στην παραπάνω διαμόρφωση, του να έχουμε την τελευταία γραμμή του πίνακα που πολλαπλασιάζει τα u συμπληρωμένη από μηδενικούς πίνακες (και φυσικά το διάνυσμα που περιέχει τα u θα συμπεριλάβει έναν ακόμη όρο , τον $u=u_n$.). Αντίστοιχα ο πίνακας E που πολλαπλασιάζει την αρχική συνθήκη x_0 θα περιλαμβάνει τον πίνακα F_N στη τελευταία γραμμή ενώ ο b θα έχει στη τελευταία γραμμή τον όρο c_n.

Τελικά για τους περιορισμούς έχουμε
σε συμπυκνωμένη μορφή : $\boldsymbol{Gu \leq Ex_0 + b}$ (5.12)
Σημείωση: από την από πάνω σχέση παρατηρούμε ότι

$G \in \mathbb{R}^{(N+1)h\times(m*n)}, E \in \mathbb{R}^{h(N+1)\times n}, b \in \mathbb{R}^{h*(N+1)\times 1}$

Παρατήρηση: Το σύνολο των περιορισμών αποτελεί ένα κυρτό πολύγωνο και αποδεικνύεται αν δούμε τις τομές των πεπερασμένων ημιχώρων που προκαλούνται από την παραπάνω ανισοϊσότητα.

Σε αυτό το σημείο θα δείξουμε πως υλοποιούμε την διαμόρφωση των πινάκων στο Matlab. Η περιγραφή του μηχανισμού είναι απαραίτητη, ούτως ώστε να γίνουν περισσότερο ευκρινή τα κομμάτια του κώδικα όπου θα μπορούσε να

υπάρξει τροποποίηση με σκοπό την βελτίωση ταχύτητας εκτέλεσης του αλγορίθμου διαμόρφωσης. Η ταχύτητα είναι ένα πολύ σημαντικό κριτήριο για την επιλογή αλγορίθμου επίλυσης (αλλά και διαμόρφωσης) τετραγωνικών προβλημάτων, ιδιαίτερα όταν εφαρμόζουμε λύσεις που συμπεριλαμβάνουν μεγάλους ορίζοντες και πολλές μεταβλητές εισόδου και κατάστασης.
Έτσι, οι πίνακες A_bar , B_bar, D_bar οι οποίοι αποτελούν τους προσαυξημένους πίνακες που αναφέραμε παραπάνω, διαμορφώνονται με τις ακόλουθες εντολές:

```
A_bar=eye(nx);%% όπου nx είναι ο αριθμός μεταβλητών κατάστασης
for i=1:N %% όπου N είναι ο ορίζοντας (τιμές θετικές ακέραιες)
    A_bar=[A_bar;A^i];
end
```

Δηλαδή θέτουμε αρχικά τον πίνακα A_bar να είναι ο μοναδιαίος με διαστάσεις όσες ο αριθμός των μεταβλητών κατάστασης. Στη συνέχεια τοποθετούμε κάτω από τον πίνακα έναν άλλο πίνακα υψωμένο στην δύναμη που αντιστοιχεί στην τιμή του δείκτη i. Η διαδικασία επαναλαμβάνεται μέχρις ότου το i πάρει όλες τις ακέραιες τιμές στο διάστημα $(1, horizon)$. Προφανώς σε αυτό το σημείο το Matlab τονίζει ότι το γεγονός ότι υπάρχει πίνακας που αλλάζει μέγεθος σε κάθε

επανάληψη, θα μειώσει την ταχύτητα εκτέλεσης. Παρόλα αυτά προτιμήθηκε αυτή η διαμόρφωση εξαιτίας της απλότητας της.

```
B_bar:
B_bar=zeros((N+1)*nx,nu*N);

for j=1:N
  Bcum=B;
  for k=j:-1:1
      B_bar(1+nx*j:nx*(j+1),1+nu*(k-1):k*nu)=Bcum;
      Bcum=A*Bcum;
  end
end
```

Εδώ η διαμόρφωση είναι περισσότερο περίπλοκη. Οι διαστάσεις καθορίζονται εξαρχής με βάση την τιμή του ορίζοντα και στη συνέχεια τοποθετούνται σε κάθε πινακογραμμή οι κατάλληλοι πίνακες .
Έτσι για έναν υποθετικό πίνακα B_bar, η πορεία που θα ακολουθηθεί με βάση τον παραπάνω κώδικα θα είναι:

$$\begin{bmatrix} 0_{nx*nu} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} B & 0 & 0 \\ 0 & . & : \\ 0 & . & : \end{bmatrix} \rightarrow \begin{bmatrix} B & 0 & 0 \\ 0 & B & : \\ 0 & 0 & : \end{bmatrix} \rightarrow \begin{bmatrix} B & 0 & 0 \\ A*B & B & : \\ 0 & 0 & 0 \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} B & 0 & 0 \\ A*B & B & 0 \\ 0 & 0 & B \end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix} B & 0 & 0 \\ AB & B & 0 \\ AAB & AB & B \end{bmatrix}$$

Ο σύνθετος πίνακας D_bar κατασκευάζεται και αυτός με παρόμοιο τρόπο με την μόνη διαφορά να βρίσκεται στις διαστάσεις των επιμέρους πινάκων (όπου αυτή την φορά είναι σίγουρα τετραγωνικοί, σε αντίθεση με τον B_bar όπου οι επιμέρους πίνακες δεν είναι απαραίτητο να είναι τετραγωνικοί.

Οι εντολές για την διαμόρφωση του D_bar είναι οι ακόλουθες:

```
D_bar=zeros((N+1)*nx,nx*N);
for j=1:N
  Dcum=eye(nx);
  for k=j:-1:1
      D_bar(1+nx*j:nx*(j+1),1+nx*(k-1):k*nx)=Dcum;
      Dcum=A*Dcum;
  end
end
```

Επίσης έχουμε και τους πίνακες Q_bar, R_bar,S_Bar,q_bar,r_bar,s_bar. Αυτοί οι πίνακες συμπεριλαμβάνονται στην αντικειμενική συνάρτηση και έχουν ρόλο βαθμονόμησης. Γενικά συνηθίζεται είτε να είναι σύνθετοι διαγώνιοι μοναδιαίοι πίνακες ή διανύσματα ή αντίστοιχα μηδενικοί. Για την δημιουργία των σύνθετων πινάκων αυτών, επειδή έχουν μορφή

$$S_{bar} = \begin{bmatrix} S & 0 & 0 \\ 0 & \ddots & : \\ : & 0 & S_N \end{bmatrix}, Q_{bar} = \begin{bmatrix} Q & 0 & 0 \\ 0 & \ddots & : \\ 0 & 0 & Q_{N+1} \end{bmatrix}, r_{bar} = [r^T \quad \dots \quad r_N^T],$$

$$s_{bar} = [s \quad \dots \quad s_{N+1}], q_{bar} = [q^T \quad \dots \quad q_{N+1}^T]$$

Χρησιμοποιούμε τις ακόλουθες εντολές του Matlab:

```
Q_bar= kron(eye(N+1),Q);
R_bar= kron(eye(N),R);%
S_bar= kron(eye(N),S);%initial was N
S_bar= [S_bar; zeros(nx,nu*N)];   %need to augment with zero rows to give
```

```
proper dimensions

q_bar=kron(ones(1,N+1),q');
r_bar=kron(ones(1,N),r');
s_bar=kron(ones(1,N+1),s);


f_bar=kron(ones(N,1),f);
```

Όσον αφορά τους πίνακες των περιορισμών έχουμε:

```
%Creation of matrix G in constraints
G=zeros(N*size(Gk,1),size(Gk,2)*(N));
%nu=number of inputs
powerA=0;
metritis_grammwn=1;
metritis_stilwn=1;
for j=1:size(Gk,2):size(G,2)
    for i=1:size(Gk,1):size(G,1)
        if metritis_grammwn==metritis_stilwn
           G(i:size(Gk,1)+i-1,j:size(Gk,2)+j-1)=Gk;
        elseif metritis_grammwn>metritis_stilwn
           G(i:size(Gk,1)+i-1,j:size(Gk,2)+j-1)=Fk*(A^powerA)*B;
           powerA=powerA+1;
        else

           G(i:size(Gk,1)+i-1,j:size(Gk,2)+j-1)=zeros(size(Gk));
        end
        metritis_grammwn=metritis_grammwn+1;
    end
    powerA=0;
    metritis_grammwn=1;
    metritis_stilwn=metritis_stilwn+1;
end
```

Εδώ ο πίνακας G διαμορφώνεται κάπως διαφορετικά ενώ θα μπορούσε να γίνει με αντίστοιχο τρόπο με την δημιουργεία του B_bar,D_bar. Ελέγχοντας τους χρόνους διαμόρφωσης των 2 πινάκων με τις εντολές tic&toc βλέπουμε ότι ο χρόνος για την δημιουργία του πίνακα $B_{bar} \in \mathbb{R}^{10\times 8}$ $με$ $\{horizon = 4, \#states = 2, \#inputs = 2\}$ $είναι$ $ίσος$ $με$

$t_{B_{bar}} = 0.08sec$

$ενώ$ $για$ $τον$ $πίνακα$ $G \in \mathbb{R}^{32\times 8}$ $για$ $το$ $ίδιο$ $πρόβλημα$ $αναφοράς$ $έχουμε$

$$t_{G_{constraints}} = 0.01sec$$

Έτσι ενώ ο κώδικας για τον πίνακα G είναι πολυπλοκότερος, είναι ταυτόχρονα πιο γρήγορος.

Η πορεία που ακολουθεί η πλήρωση του G φαίνεται παρακάτω:

$$\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix} \rightarrow \begin{bmatrix}G_k & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix} \rightarrow \begin{bmatrix}G_k & 0 & 0\\ F_kB & 0 & \vdots\\ 0 & 0 & 0\end{bmatrix} \rightarrow \begin{bmatrix}G_k & 0 & 0\\ F_kB & 0 & \vdots\\ F_kAB & 0 & 0\end{bmatrix} \rightarrow$$

$$\begin{bmatrix}G_k & 0 & 0\\ F_kB & G_k & 0\\ F_kAB & 0 & 0\end{bmatrix} \rightarrow [\dots] \rightarrow \begin{bmatrix}G_k & 0 & 0\\ F_kB & G_k & 0\\ F_kAB & F_kB & G_k\end{bmatrix}$$

Για τους πίνακες E,b εξαιτίας της απλότητας των εντολών, χρησιμοποιήθηκε η λογική που ακολουθήθηκε στη δημιουργία του A_bar.

Τέλος για να φτάσουμε από την μορφή

$$x(t) = Ax_{t-1} + B * u_{t-1} + Df$$

$$\text{Subject to : } G_k u_t \leq -F_k x_t + c_k$$

στην τετραγωνική πρωτεύουσα μορφή:

$$V^*(x_0) = \frac{1}{2}u^T Mu + (Cx_o + g)^T u + \frac{1}{2}x_o^T Y x_0$$

$$s.t\ Gu < Ex_0 + b$$

Θέτουμε με βάση τους παραπάνω υπολογισμούς:

```
M=1/2*B_bar'*Q_bar*B_bar+B_bar'*S_bar+1/2*R_bar;
C=B_bar'*Q_bar*A_bar+S_bar'*A_bar;
g=B_bar'*Q_bar*D_bar*f_bar+S_bar'*D_bar*f_bar+B_bar'*
q_bar'+r_bar';
```

Έτσι αφού διαμορφώθηκαν οι πίνακες όπως αναφέρονται στο [Bemporad, Patrinos 2012] θα συνεχίσουμε γράφοντάς τους στην τελική μορφή όπως συνηθίζεται να συναντάται στην βιβλιογραφία. Δηλαδή τελικά το πρωτεύων Q.P θα είναι της μορφής:

$$V^*(x_0) = \frac{1}{2}u^T Mu + Bu + \frac{1}{2}x_0^T Y x_0$$

s.t $$Au \leq b$$

Προφανώς θα ισχύει: $B = (Cx_o + g)^T$, $A = G$, $b = Ex_0 + b$

Στη συνέχεια της εργασίας αυτής θα δούμε και θα συγκρίνουμε την απόδοση των αλγορίθμων GPAD,QUADPROG,CPLEX σε 2 benchmark προβλήματα και σε ένα κατασκευασμένο από εμάς πρόβλημα όπου συνδυάζονται οι κώδικες που γράψαμε παραπάνω με τον κώδικα που λύνει το πρόβλημα. Στα αποτελέσματα περιλαμβάνουμε τις λύσεις για τους default αλγορίθμους που ακολουθούν οι CPLEX και QUADPROG . Συγκεκριμένα όσων αφορά το QUADPROG, αυτό εμφανίζει μήνυμα που λέει ότι θα χρησιμοποιήσει active set αλγόριθμο. Ο κώδικας που θα χρησιμοποιηθεί για το τεχνητό πρόβλημα καθώς και για τα benchmark επισυνάπτεται παρακάτω στο παράρτημα και η ανάλυσή του θα γίνει στο diesel πρόβλημα:

6 Αποτελέσματα

6.1 Εφαρμογή 1η : Τεχνικά κατασκευασμένο πρόβλημα:

Θα θεωρήσουμε το ακόλουθο πρόβλημα:

$$\begin{bmatrix} x_{11} \\ x_{12} \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_{01} \\ x_{02} \end{bmatrix} + \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_{01} \\ u_{02} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{6.1}$$

επίσης όσων αφορά την συνάρτηση κόστους (αντικειμενική συνάρτηση) έχουμε:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad S = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad q = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad r = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad s = [0]$$

$$F_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \; G_k = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}, \; c_k = \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \end{bmatrix}, F_N = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}, c_n = \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \end{bmatrix} \tag{6.2}$$

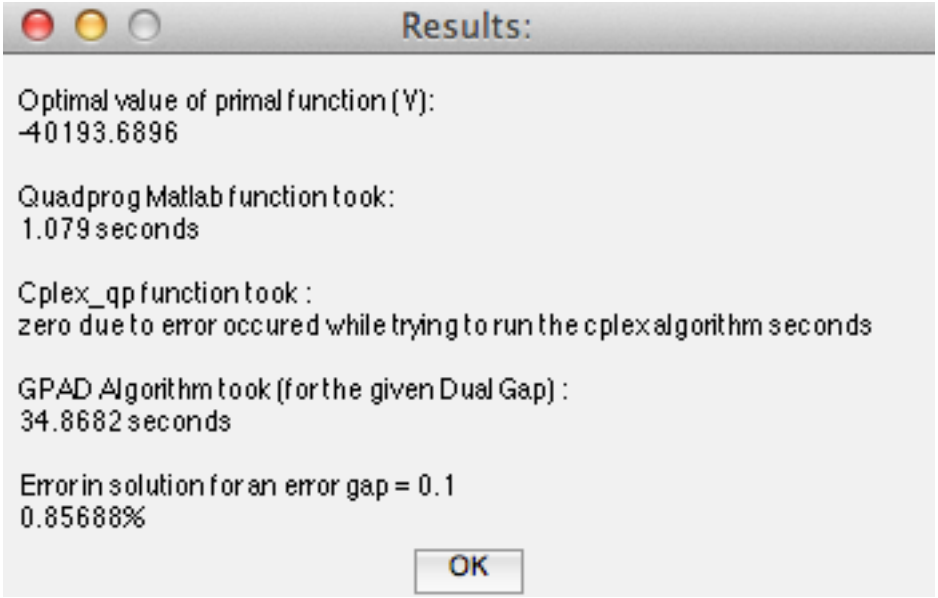
Θα λύσουμε το πρόβλημα κάνοντας την διαμόρφωση των πινάκων M, $(Cp + g)^T$, $G, Ep + b \quad (p = x_0)$
κατά τρόπο που αναφέρθηκε παραπάνω. Η αντιστοιχία με τους πίνακες όπως ορίστηκαν στον κώδικα γίνεται με τις εντολές:

```
HQ=M;%M Matrix

fQ=(C*Xo+g)';%this is noted as (Cp+g)' in our paper
AQ=G;%this is noted as G in our paper
%bQ=[lbA(1,:)';-ubA(1,:)';lb(1,:)';-ub(1,:)'];%this is noted as Ep-b in our
paper
bQ=E*Xo+b;
```

Αποτελέσματα:

Λύνοντας το πρόβλημα για N=4 παίρνουμε την λύση:

$$Quadprogsolution = \begin{bmatrix} -2.9733 \\ -3.3719 \\ -0.8218 \\ 0.4889 \\ 3.0169 \\ 3.1875 \\ -10 \\ -10 \end{bmatrix}, GPAD\ solution = \begin{bmatrix} -2.9749 \\ -3.3773 \\ -0.8288 \\ 0.4886 \\ 3.0220 \\ 3.1672 \\ -10.0544 \\ -10.0162 \end{bmatrix}$$



Παρατήρηση: Εδώ το GPAD φαίνεται να έχει πολύ μεγάλο χρόνο επίλυσης και για αρκετά μεγάλο δυικό κενό. Αυτό ερμηνεύεται στο γεγονός του ότι όταν αυξήσουμε παραδείγματος χάρη τον ορίζοντα, υπολογίζοντας τον αντίστροφο M^T παίρνουμε μήνυμα ότι ο πίνακας είναι badly scaled άρα τα αποτελέσματα μπορεί να είναι ανακριβή. Επομένως έτσι εξηγείται η δυσκολία του αλγορίθμου να συγκλίνει στην λύση.

6.2 Εφαρμογή $2^{η}$:Εφαρμογή του αλγορίθμου GPAD στο πρόβλημα Diesel:

Το benchmark αυτό πρόβλημα αντλήθηκε από το site: kuleuven.be
Αυτό το πρόβλημα περιέχει 20 μεταβλητές, φραγμένες, 20 ανισοτικούς περιορισμούς και 0 ισοτικούς περιορισμούς. Στόχος είναι ο έλεγχος της πραγματικής απευθείας εισροής σε ένα turbo Diesel κινητήρα. Τα δεδομένα προέρχονται από προσομοιώσεις κλειστού βρόγχου και χρησιμοποιώντας προσεγγιστικό γραμμικό μοντέλο γύρω από μια περιοχή λειτουργίας 2100-2500 rpm ενώ η μάζα του εισερχόμενου καυσίμου ήταν στα $0 - 30\frac{mg}{stroke}$. Η ταχύτητα της μηχανής καθώς και το ποσό του εισερχόμενου καυσίμου διατηρήθηκαν σταθερά στις 2300 στροφές και $15\frac{mg}{stroke}$ αντίστοιχα ενώ ο ρυθμιστής παρκολουθούσε 2 βηματικές αλλαγές στα set points της μαζικής παροχής και της εισερχόμενης μανομετρικής πίεσης. Οι είσοδοι είναι οι ρυθμοί κίνησης της βαλβίδας που σχετίζεται με την ρύθμιση της ανακύκλωσης του εξερχόμενου αερίου, καθώς και η θέση της μεταβλητής γεωμετρίας του turbocharger. Κάτω και πάνω φράγματα τέθηκαν σε όλες τις μεταβλητές εισόδου. Ο ορίζοντας ελέγχου είναι ίσος με 10 διαστήματα (κάθε ένα μήκους 50ms) και ο ορίζοντας πρόβλεψης επιλέχθηκε να ισούται με 4s για ρυθμό προσομοίωσης ίσο με 50ms.

Λύση:
Το πρώτο τμήμα της λύσης περιλαμβάνει την εισαγωγή των δεδομένων στο Matlab και ο κώδικας ήταν έτοιμος από την ιστοσελίδα. Πέρα από τα δεδομένα του προβλήματος, δόθηκαν και οι λύσεις ώστε να μπορούμε να συγκρίνουμε τα αποτελέσματα του δικού μας αλγόριθμου.

```
%%**************************************************************
**********
```

```
%%
*
%%  ONLINE QP BENCHMARK COLLECTION
*
%%  http://homes.esat.kuleuven.be/~optec/software/onlineQP
*
%%
*
%%  maintained by: Moritz Diehl and Hans Joachim Ferreau
*
%%
*
%%*************************************************************************
***


%%
%% filename:     loadBenchmark.m
%% author:       Hans Joachim Ferreau, joachim.ferreau@esat.kuleuven.be
%%
%% description:  this script loads all benchmark data into the workspace
%%               (attention: existing data may be overridden!)


disp('INFO (loadBenchmark):  Loading benchmark data... ');

%% load dimensions
dims = load('dims.oqp');
nQP = dims(1);
nV  = dims(2);
nC  = dims(3);
nEC = dims(4);
clear dims;

%% load other data ...
H  = load('H.oqp');
g  = load('g.oqp');
lb = load('lb.oqp');
ub = load('ub.oqp');

%% ... including constraints (if any)
if ( nC > 0 )
    A   = load('A.oqp');
    lbA = load('lbA.oqp');
    ubA = load('ubA.oqp');
end

%% load optimal solutions
x_opt = load('x_opt.oqp');
y_opt = load('y_opt.oqp');
obj_opt = load('obj_opt.oqp');


%% finally, perform some consistency checks ...
successful = 1;

if ( ( nQP <= 0 ) || ( nV <= 0 ) || ( nC < 0 ) || ( nEC < 0 ) || ( nEC > nC )
)
    disp('ERROR (loadBenchmark):  Dimension data invalid!');
    return;
end

[N,M] = size(H);
if ( ( N ~= M ) || ( N ~= nV ) )
    disp('ERROR (loadBenchmark):  Hessian matrix has wrong dimensions!');
    successful = 0;
end

[N,M] = size(g);
if ( ( N ~= nQP ) || ( M ~= nV ) )
```

```
    disp('ERROR (loadBenchmark):  Gradient series has wrong dimensions!');
    successful = 0;
end

[N,M] = size(lb);
if ( ( N ~= nQP ) || ( M ~= nV ) )
    disp('ERROR (loadBenchmark):  Lower bound series has wrong dimensions!');
    successful = 0;
end

[N,M] = size(ub);
if ( ( N ~= nQP ) || ( M ~= nV ) )
    disp('ERROR (loadBenchmark):  Upper bound series has wrong dimensions!');
    successful = 0;
end

%% ... including constraints (if any)
if ( nC > 0 )
    [N,M] = size(A);
    if ( ( N ~= nC ) || ( M ~= nV ) )
        disp('ERROR (loadBenchmark):  Constraint matrix has wrong
dimensions!');
        successful = 0;
    end

    [N,M] = size(lbA);
    if ( ( N ~= nQP ) || ( M ~= nC ) )
        disp('ERROR (loadBenchmark):  Lower bound series has wrong
dimensions!');
        successful = 0;
    end

    [N,M] = size(ubA);
    if ( ( N ~= nQP ) || ( M ~= nC ) )
        disp('ERROR (loadBenchmark):  Upper bound series has wrong
dimensions!');
        successful = 0;
    end
end

if ( successful == 0 )
    disp('INFO  (loadBenchmark):  Errors occured while loading benchmark
data!');
else
    disp('INFO  (loadBenchmark):  Benchmark data loaded successfully!');
end

clear N M successful;


%end of file
```

Σε αυτό το σημείο έχει ολοκληρωθεί η καταχώρηση των δεδομένων. Ακολουθεί η κατάλληλη αντιστοίχηση σε καινούργιους πίνακες οι οποίοι θα εισαχθούν στην εντολή QUADPROG του Matlab. Ουσιαστικά το βήμα της επαναπροσδιόρισης των πινάκων είναι απαραίτητο επειδή στο αρχικό αρχείο δόθηκαν προσαυξημένοι πίνακες οι οποίοι περιέχουν στοιχεία για την λύση 600 προβλημάτων. Επειδή μας ενδιαφέρει να δούμε τους χρόνους λύσης ενός προβλήματος, απαιτείται να απομονώσουμε τα κατάλληλα τμήματα.

```
X0=x_opt(1,:);
I=eye(size(H,1));
HQ=H;
fQ=g(1,:);
AQ=[A;-A];
bQ=[ubA(1,:)';-lbA(1,:)'];
```

Η αντιστοίχηση των πινάκων ολοκληρώθηκε. Ακολουθεί η μέτρηση του χρόνου εκτέλεσης της εύρεσης της βέλτιστης λύσης:

```
t=tic;%Sets timer to Time the run of quadprog algorithm!
[X,fval]=quadprog(HQ,fQ,AQ,bQ,[],[],lb(1,:)',ub(1,:)');
elapsedquadprog=toc(t);%Records time to run quadprog function
```

Αν εκτελέσουμε τις παραπάνω εντολές και συγκρίνουμε το αποτέλεσμα με την λύση που δίνεται από το site, παρατηρούμε ότι είναι ταυτόσημες.
Ακολουθεί ο έλεγχος χρόνου για το Cplex.

Σημείωση: Το Cplex απαιτεί ο Η να είναι συμμετρικός πέρα από θετικά ημιορισμένος επομένως προσθέσαμε ένα τμήμα κώδικα που φροντίζει να είναι 100% συμμετρικός χωρίς την παραμικρή απόκλιση.

```
%% we have to make sure H_cplex matrix is exactly symmetric so that cplex
won't crash.
for i=1:size(HQ,1)
    for j=i+1:size(HQ,1)
        HQ(i,j)=HQ(j,i);
    end
end
%% CPLEX QP PERFORMANCE:
try
    t=tic;
    [cplex_solutin,fval_cplex]=cplexqp(HQ,fQ',AQ,bQ,[],[],lb(1,:)',ub(1,:)');
    elapsed_cplex=toc(t);%Records time to run quadprog function
catch
 disp('an error occured while trying to run Cplex algorithm....check if u
have 32 bit of matlab or cplex algos installed')
    elapsed_cplex='zero due to error occured while trying to run the cplex
algorithm';
end
```

Εδώ ολοκληρώθηκε και η λύση με το Cplex η οποία είναι επίσης ταυτόσημη με την λύση που δίνει η ιστοσελίδα.
Σημείωση: Προστέθηκε ένα τμήμα κώδικα που να εξασφαλίζει ότι ολόκληρος ο κώδικας θα τρέξει ακόμη και αν δεν υπάρχει το Cplex στο path του Matlab. Φυσικά σε μια τέτοια περίπτωση, στο τέλος της εκτέλεσης θα εμφανιστεί μήνυμα ότι δεν χρησιμοποιήθηκε ο αλγόριθμος Cplex.

Ακολουθεί ο κώδικας που φτιάχτηκε για να υλοποιεί τον αλγόριθμο GPAD

Βήμα 1: Ορισμός των πινάκων σημειώνοντας δίπλα την αντιστοιχία τους με τους πίνακες που αναγράφονται στην δημοσίευση [Bemporad, Patrinos 2012]

```
%% GPAD algorythm-Declaration of Matrices
%Checking time and results which will be displayed plus given duality gap

X0=x_opt(1,:);
I=eye(size(H,1));
HQ=H;%M Matrix

fQ=g(1,:);%this is noted as (Cp+g)' in our paper
AQ=[A;-A;I;-I];%this is noted as G in our paper
%bQ=[lbA(1,:)';-ubA(1,:)';lb(1,:)';-ub(1,:)'];%this is noted as Ep-b in our
paper
bQ=[ubA(1,:)';-lbA(1,:)';ub(1,:)';-lb(1,:)'];
```

Ολοκληρώθηκε η δήλωση των πινάκων που αφορούν το πρωτεύων πρόβλημα και ακολουθεί η δήλωση αρχικών τιμών για τις δυικές και πρωτεύουσες μεταβλητές

```
y=zeros(size(bQ,1),1);
y_minus1=zeros(size(y,1),1);
z_v=zeros(nV,1);
Counter=0;
```

Δημιουργούμε τους πίνακες που αφορούν το δυικό πρόβλημα όπως έχουμε αναφέρει στο κεφάλαιο που αναλύει την διαμόρφωση του αλγορίθμου:

```
MINV=inv(HQ);

g_p=MINV*fQ';
M_g=MINV*AQ';
L=max(eig(HQ));
G_L=AQ/L;
p_D=-(bQ)/L;
theta=[1,1];%as defined in(8e)
beta=theta(2)*(theta(1).^(-1)-1);%theta(2)is the most recent whereas
theta(1)is one iteration before theta(1)
% theta(1)=theta(2);
% theta(2)=(sqrt(theta(1)^4+4*theta(1)^2)-theta(1)^2)/2;
%V=Dualfi(HQ,MINV,fQ,AQ,bQ,y,z_v);
z_vminus1=z_v;
```

Τώρα ακολουθεί ο υπολογισμός ενός εκ των δύο κριτηρίων τερματισμού, δηλαδή τον υπολογισμό του δυικού κενού (dual gap).

```
%% Calculations which will be used to find the sum of Dual and Primal
function
D_mod=AQ*MINV*fQ'+bQ;%Dp +d matrix component
D_mod=D_mod';
d_p=(1/2)*fQ*MINV*fQ';
H_moded=AQ*MINV*AQ';
```

Η παρακάτω συνάρτηση αποτελεί τον υπολογισμό του αθροίσματος της δυικής και της πρώιμης συνάρτησης για τις τωρινές τιμές των δυικών και πρωτευουσών

μεταβλητών που έχουμε υπολογίσει μέχρι τώρα αντίστοιχα:

```
V=Dualfotos(HQ,H_moded,d_p,D_mod,fQ,y,z_v);
```

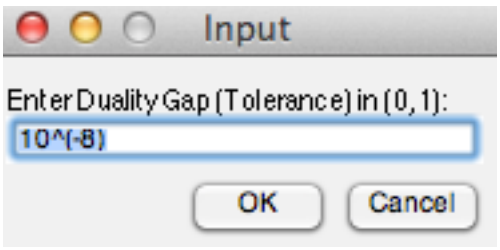
{Σημειώνεται ότι στο Matlab έχουμε δημιουργήσει ένα ξεχωριστό m.file με τον τύπο:

```
function V=Dualfotos(HQ,H_moded,d_p,D_mod,fQ,y,z_v)
%takes input vector y size(size(bQ,1),1) and the defining matrices
%calculates output of dual equivalent
%Reminder:HQ<->M,fQ<->(Cp+g)',G<->AQ,

z_v=-z_v;
V=(1/2)*(y'*H_moded*y+z_v'*HQ*z_v)+fQ*z_v+D_mod*y+d_p;
end}
```

Το επόμενο κομμάτι κώδικα αφορά τα παράθυρα που εμφανίζονται στον χρήστη και τον ρωτούν το δυικό κενό για το οποίο επιθυμεί να λυθεί το πρόβλημα, αν επιθυμεί να λάβει διαγράμματα για την απόδοση του αλγορίθμου, όπως πχ τις επαναλήψεις και τον χρόνο, για κάποιες τιμές δυικού κενού έως ότου φτάσει στο δυικό κενό που δήλωσε στο προηγούμενο ερώτημα. Έτσι έχουμε:

```
%% FANCY INPUT WINDOWS!
prompt = {'Enter Duality Gap (Tolerance) in (0,1):'};
dlg_title = 'Input';
num_lines = 1;
def = {'10^(-8)'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
R=cell2mat(answer);
R=str2num(R);
```

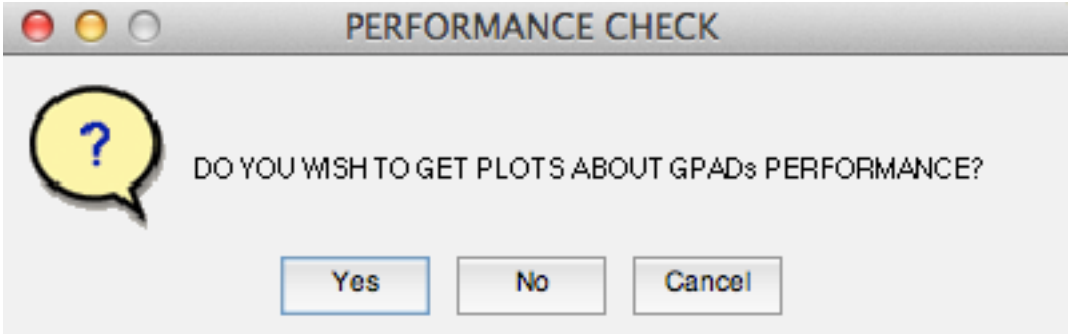
Το οποίο εμφανίζει στον χρήστη το μήνυμα:




```
prompt = {'DO YOU WISH TO GET PLOTS ABOUT GPADs PERFORMANCE?'};
dlg_title = 'PERFORMANCE CHECK';
Check_to_iterate= questdlg(prompt,dlg_title);
if (R<0||R>1)&&strcmp(Check_to_iterate,'Yes')
    R=10^(-5);
end
if strcmp(Check_to_iterate,'Yes')&&R<1&&R>0
    End_of_iterations=-log10(R);
    m=1;%will be used to record the values of counter,time, etc
    step=0.1;
else
```

```
    End_of_iterations=1;
    step=1;
    m=1;
end
```

Το παραπάνω κομμάτι του κώδικα εμφανίζει στον χρήστη το μήνυμα:



Ακολουθούν κάποιες εντολές οι οποίες θα μας βοηθήσουν να κρατάμε κάποια στοιχεία κατά την εκτέλεση του αλγορίθμου όπως για παράδειγμα την δημιουργία διανύσματος κατάλληλων διαστάσεων όπου θα αποθηκεύεται ο χρόνος εκτέλεσης για κάθε δυικό κενό (στην περίπτωση που ο χρήστης επιθυμεί να πάρει τέτοια διαγράμματα) ή την δημιουργία διανύσματος που θα περιέχει τον αριθμό επαναλήψεων που έγιναν ώστε να λυθεί το πρόβλημα για κάποιο δυικό κενό αντίστοιχα.

```
%%Dimensions for recording data on algorithms performance
eg_vector=zeros(size(1:step:End_of_iterations,2),1);
elapsed_vector=eg_vector;
Counter_vector=eg_vector;
errorvector_tracker=eg_vector;


for i=1:step:End_of_iterations
    if strcmp(Check_to_iterate,'Yes')
        eg=10.^(-i);
    eg_vector(m)=eg;
    else
        eg=R;
    end
    %% I CONNECT THIS PART WITH THE QUADPROG FORMULATION
```

Σε αυτό το σημείο όλες οι παράμετροι έχουν τεθεί, και είναι το σημείο όπου μπορούμε να ξεκινήσουμε να μετρούμε τον χρόνο εκτέλεσης του κώδικα (άρα και του αλγορίθμου κατά ένα έμμεσο τρόπο).

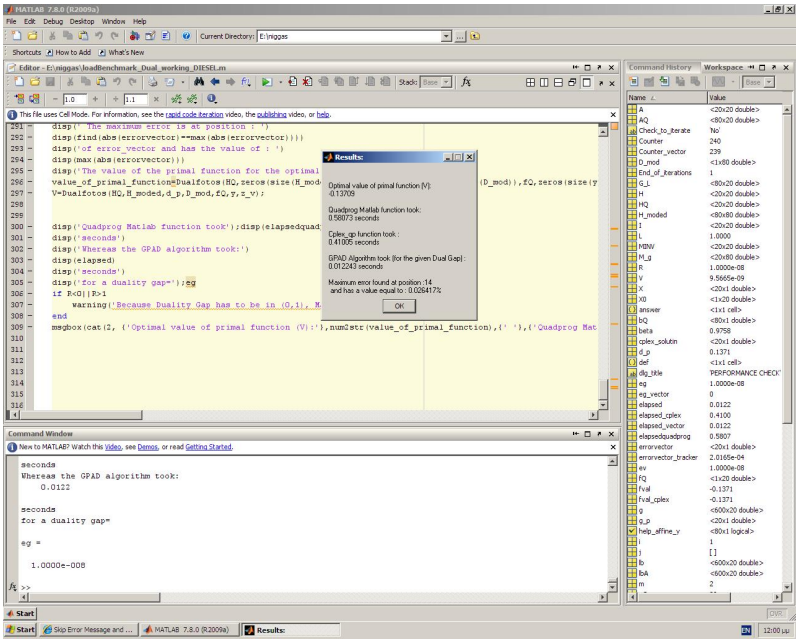
```
tonos=tic;%Sets the Timer to calculate time it takes for the GPAD algorithm
to run


%% Application:(specify eg-tolerance- evgap)
ev=eg;
while (sum(gt(AQ*(-z_v)-bQ,eg)))~=0||(gt(V,ev))~=0
    %step1:find w,z,y(counter+1)

    w_v=y+beta*(y-y_minus1);
    z_app=-M_g*w_v-g_p;%approximation of primal
    if Counter>0
        y_minus1=y;
    end
    y=(w_v+G_L*z_app+p_D);%Dual solution

    help_affine_y=(y<0);%variable that helps make all <0 elements
    %equal to 0
    y(help_affine_y)=0;%all negative values become zero

    V=Dualfotos(HQ,H_moded,d_p,D_mod,fQ,y,z_v);
    %V=Dualfi(HQ,MINV,fQ,AQ,bQ,y,z_v);
    %  Calculation of terms that define Z_v(our primal solution)
    if Counter>0
     z_v=(1-theta(2))*z_v-theta(2)*z_app;%Primal solution
    end

    %Calculations needed for the next iteration:
    theta(1)=theta(2);
    theta(2)=(sqrt(theta(2)^4+4*theta(2)^2)-theta(2)^2)/2;

    assert(theta(2)~=theta(1),'SAME THETAS')%CONSISTENCY CHECK
    beta=theta(2)*(theta(1).^(-1)-1);
    y_minus1=y;
    %Break-Loop-Condition
    Counter=Counter+1;
%     if Counter>500
%         disp('Note: Counter exceeded 500 iterations')
%         break;
%     end
    Counter_vector(m)=Counter;
    Counter=Counter+1;


end
elapsed=toc(tonos); %Time GPAD algorithm needs
elapsed_vector(m)=elapsed;
errorvector=-z_v-X0';%calculating errors for given dual gap
errorvector_tracker(m)=max(abs(errorvector));

m=m+1;


end
```

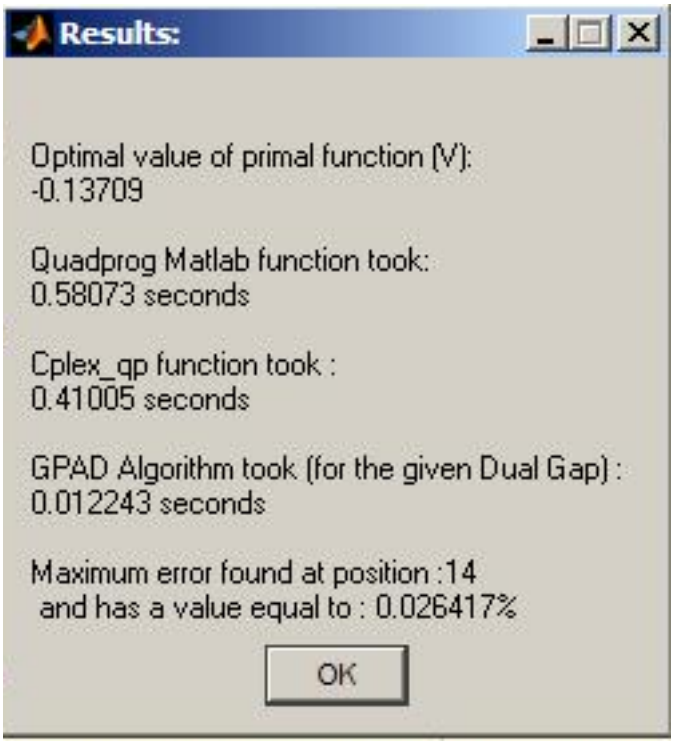
Εδώ ολοκληρώθηκε ο κώδικας του αλγορίθμου GPAD και οι ακόλουθες εντολές φροντίζουν να εμφανιστούν στον χρήστη τα αποτελέσματα της σύγκρισης:

```
if strcmp(Check_to_iterate,'Yes')&&m~=2%check for plotting

    subplot(3,1,1)
    loglog(eg_vector,elapsed_vector,'r+:')
    xlabel('Duality gap tollerence')
    ylabel('Time in seconds')
    subplot(3,1,2)
    loglog(eg_vector,Counter_vector)
```

```
    xlabel('Tollerance')
    ylabel('N. of Iterations')
    subplot(3,1,3)
    loglog(eg_vector,errorvector_tracker,'mx')
    xlabel('Tollerance')
    ylabel('Error (maximum) ')

end
disp('The optimal vector is : ')
optimum_GPAD=-z_v;
errorvector=((optimum_GPAD-X0')./X0')*100;%calculating the percentage
difference of errors
disp(' The maximum error is at position : ')
disp(find(abs(errorvector)==max(abs(errorvector))))
disp('of error_vector and has the value of : ')
disp(max(abs(errorvector)))
disp('The value of the primal function for the optimal vector is : ')
value_of_primal_function=Dualfotos(HQ,zeros(size(H_moded)),zeros(size(d_p)),z
eros(size(D_mod)),fQ,zeros(size(y)),z_v)
V=Dualfotos(HQ,H_moded,d_p,D_mod,fQ,y,z_v);


disp('Quadprog Matlab function took');disp(elapsedquadprog)
disp('seconds')
disp('Whereas the GPAD algorithm took:')
disp(elapsed)
disp('seconds')
disp('for a duality gap=');eg
if (R<0||R>1)&&strcmp(Check_to_iterate,'Yes')
    warning('Because Duality Gap has to be in (0,1), and you asked for plots,
Matlab run for the Default Gap')
end
msgbox(cat(2, {'Optimal value of primal function
(V):'},num2str(value_of_primal_function),{' '},{'Quadprog Matlab function
took: '},[num2str(elapsedquadprog) ' seconds'],{' '},{'Cplex_qp function took
: '},[num2str(elapsed_cplex) ' seconds'],{' '},{'GPAD Algorithm took (for the
given Dual Gap) :'},[num2str(elapsed) ' seconds'],{' '},{['Error in solution
for an error gap = ' num2str(R)]}, [num2str(max(abs(errorvector)))
'%']),'Results:')
```

Αποτελέσματα:



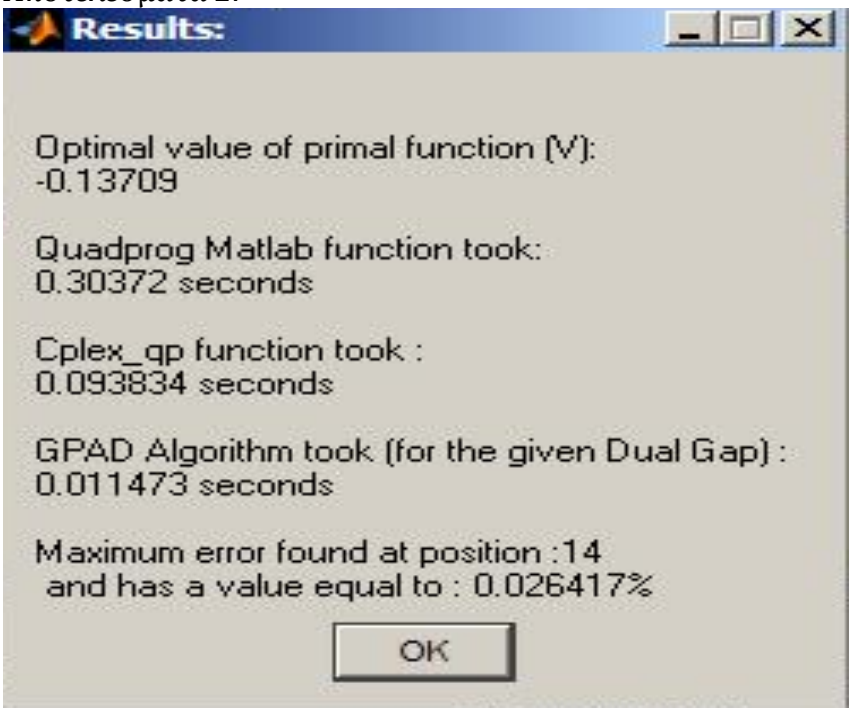


<u>Παρατήρηση:</u>

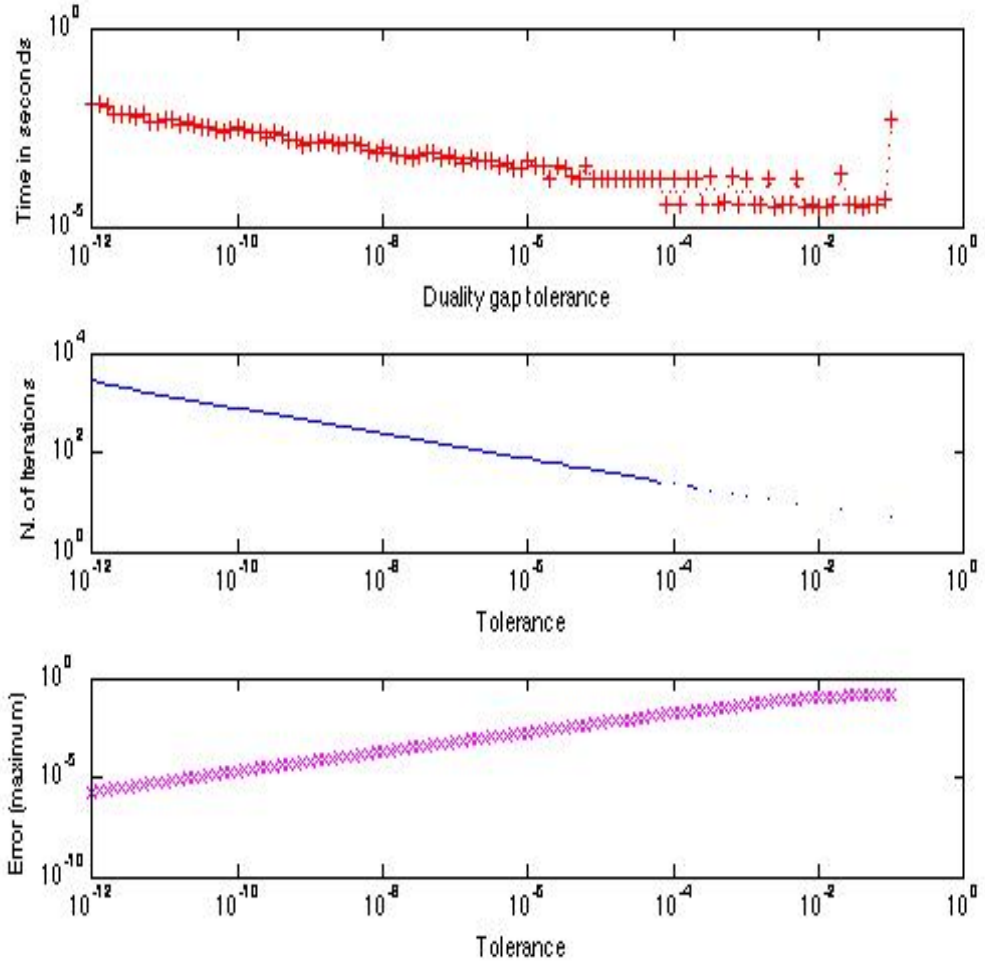
Ο αλγόριθμος GPAD είναι περίπου 97% πιο γρήγορος από το CPLEX και περίπου 98% πιο γρήγορος από το Quadprog του Matlab.

Έλεγχος 2ος : Ρύθμιση του αλγορίθμου του Quadprog από active-set σε interior point. Αντίστοιχα, αλλαγή του default αλγορίθμου του cplexqp σε Simplex

Αποτελέσματα 2:



Παρατήρηση: Έχουμε μεγάλη βελτίωση στους χρόνους εκτέλεσης των Quadprog και Cplex με την αλλαγή παραμέτρων που κάναμε. Συγκεκριμένα, ο χρόνος εκτέλεσης του Quadprog μειώθηκε κατά 48% ενώ του Cplex μειώθηκε κατά 78%. Ακολουθούν τα αποτελέσματα σχετικά με τους χρόνους εκτέλεσης για ένα πεδίο δυικού κενού από $10^{-1} - 10^{-12}$



Σχήμα 9: Το πρώτο διάγραμμα δείχνει την μεταβολή του χρόνου εκτέλεσης του κώδικα σε συνάρτηση με την ανεκτικότητα στο δυικό κενό. Το δεύτερο διάγραμμα δείχνει αντίστοιχα την σχέση $N_{iterations} = f(duality - gap)$
και το τρίτο δείχνει την μεταβολή του σφάλματος σε σχέση με την βέλτιστη λύση ώς συνάρτηση του δυικού κενού

Σημείωση: Αν προσπαθούσαμε να λύσουμε το πρόβλημα για μεγαλύτερο δυικό κενό, ο χρόνος εκτέλεσης αυξανόταν από κλάσματα του δευτερολέπτου σε ώρες και οι επαναλήψεις ξεπερνούσαν το εκατομμύριο.

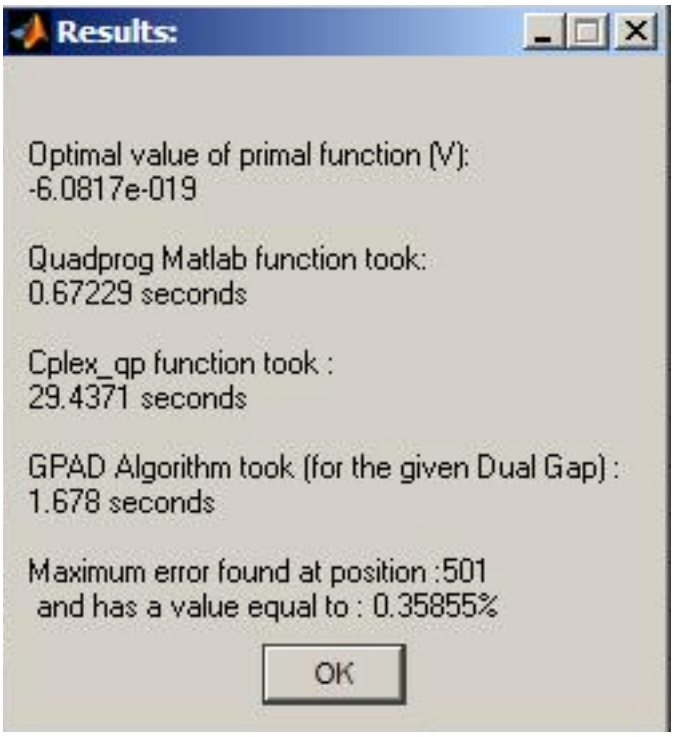
6.3 Εφαρμογή 3η :Εφαρμογή του αλγορίθμου GPAD στο πρόβλημα Distillation:

Το συγκεκριμένο πρόβλημα είναι το μεγαλύτερο διαθέσιμο benchmark πρόβλημα στο site kuleuven.be. Αποτελεί και αυτό όπως το diesel, ένα πρόβλημα MPC όπου περιέχει 800 μεταβλητές φραγμένες, 800 περιορισμούς και καθόλου ισοτικούς περιορισμούς. Αυτό το πρόβλημα είναι μια αποστακτική μονάδα πετρελαίου με 32 εισόδους, 90 εξόδους και 252 μεταβλητές κατάστασης.
Οι είσοδοι και οι έξοδοι έχουν κανονικοποιηθεί και μόνο οι είσοδοι πρέπει να ικανοποιούν περιορισμούς φραξίματος. Μόνο 4 έξοδοι που αντιπροσωπεύουν την ποιότητα των παραπροϊόντων της μονάδας έχουν επιθυμητά set points. τα οποία αποτελούν τον στόχο στην δημιουργία «πλάνου». Η απόκριση του «πραγματικού» εργοστασίου προσομοιώνεται με το να προστίθενται στην απόκριση του μοντέλου, μη μετρήσιμες τυχαίου βήματος διαταραχές στην σύνθεση του πετρελαίου, στην ποιότητα του καυσίμου αερίου, στην πίεση ατμού κορυφής και ένας κανονικά κατανεμημένος θόρυβος εξόδου.
Ο ορίζοντας πρόβλεψης και ο ορίζοντας ελέγχου έχουν το ίδιο μήκος 25 διαστημάτων με χρόνο προσομοίωσης ίσο με ένα λεπτό.

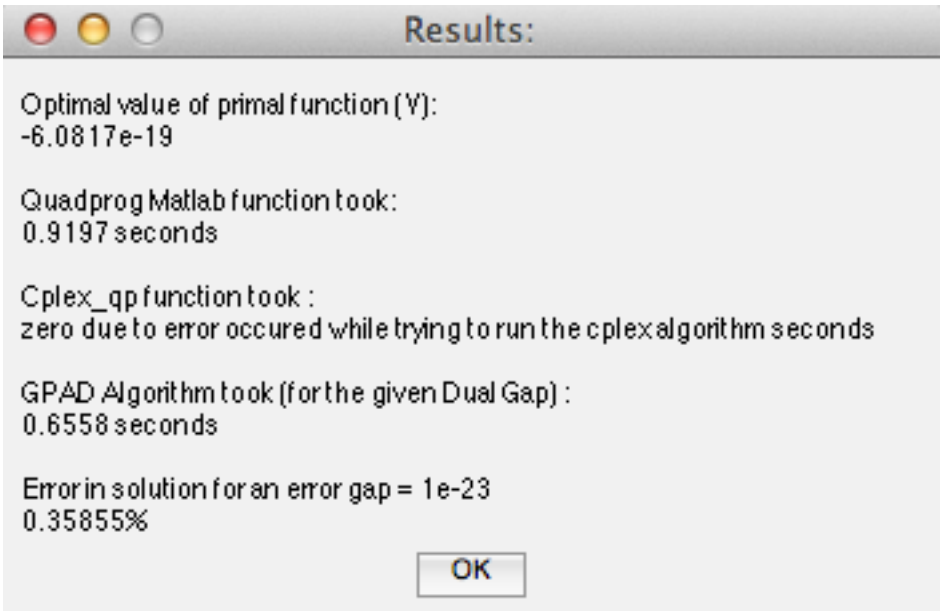
Ακολουθούν τα αποτελέσματα της επίλυσης του προβλήματος με το Quadprog,Cplex και GPAD:

Αποτελέσματα:

(για τις προεπιλεγμένες παραμέτρους των Quadprog,Cplex)



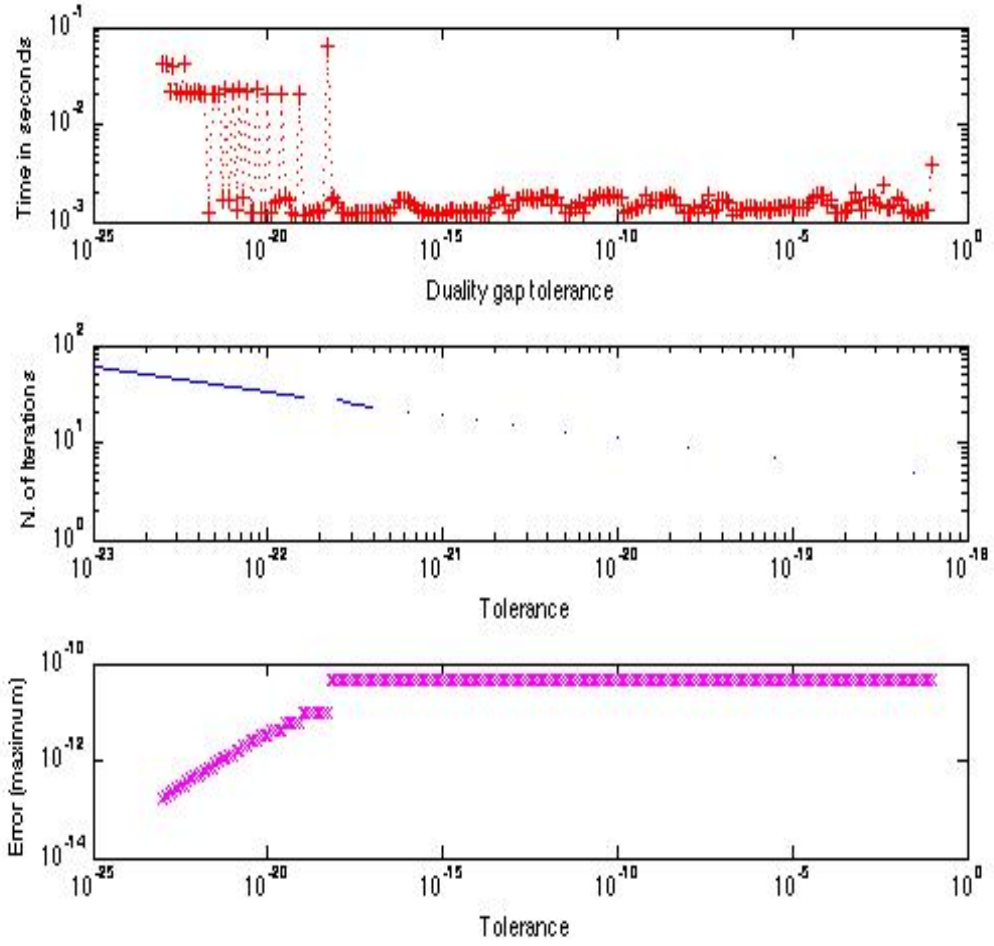
Δοκιμή σε διαφορετικό υπολογιστή(Mac 2.5GH , intel core i5, memory 4 GB 1600MHz DDR3) χωρίς Cplex:



Σημείωση: Η δοκιμή που έγινε για τροποποιημένες παραμέτρους των αλγορίθμων σε αυτό το πρόβλημα δεν επηρέασε τους χρόνους παρά μόνο για ποσοστό < 5% στον αλγόριθμο του Cplex ενώ ο χρόνος του Quadprog αυξήθηκε

κατά 10% επομένως δεν συμπεριλαμβάνουμε στην εργασία τα αποτελέσματα αυτά.

Απόδοση GPAD σε συνάρτηση με το προεπιλεγμένο δυικό κενό:



Παρατηρήσεις:

Το GPAD πάλι είναι ο πιο γρήγορος τρόπος εκτέλεσης με εξαίρεση την περίπτωση όπου το πείραμα έγινε σε υπολογιστή της σχολής (ώστε να γίνει εκτέλεση του αλγορίθμου Cplex) και εκεί χρησιμοποιήθηκε παλαιότερη έκδοση του Matlab (2009a) σε αντίθεση με το 2ο πείραμα που έγινε χωρίς το Cplex και έγινε στην r2011a έκδοση του Matlab. Μία σημαντική παρατήρηση παρόλα αυτά είναι ότι το Cplex σε αυτό το πρόβλημα φαίνεται να είναι πολύ αργό. Αυτό οφείλεται στο ότι ο default αλγόριθμος που χρησιμοποιείται από την συνάρτηση του Cplex που λύνει τετραγωνικά προβλήματα είναι αλγόριθμος εσωτερικού σημείου, επομένως τυχαίνει η δομή της Hessian μήτρας να μην είναι τέτοια ώστε να αξιοποιηθεί από ρουτίνες γραμμικής άλγεβρας.

7. Συμπεράσματα-Σύνοψη:

Τα προβλήματα στον σχεδιασμό συστημάτων, ανάλυσης δεδομένων και στατιστικής καθώς και οικονομικής διαχείρισης μπορούν συχνά να εκφραστούν ως μαθηματικά προβλήματα βελτιστοποίησης και υπάρχουν διάφορες τεχνικές για την αντιμετώπισή τους, που εξαρτώνται από τη δομή των προβλημάτων (όπως για παράδειγμα την κυρτότητά τους η οποία μας εξασφαλίζει το ότι τα τοπικά βέλτιστα, είναι παγκόσμια).

Ο αλγόριθμος GPAD αποδείχθηκε ταχύτατος στην επίλυση τετραγωνικών προβλημάτων κατά την εφαρμογή MPC. Μάλιστα ενώ η θεωρία προϋποθέτει ότι ο πίνακας στον τετραγωνικό όρο θα είναι θετικά ορισμένος, ο αλγόριθμος απέδωσε με μεγάλη επιτυχία στα προβλήματα benchmark παρότι οι πίνακες που δόθηκαν ήταν θετικά ημιορισμένοι. Για την ακρίβεια στο πρόβλημα Diesel ήταν περίπου 97.5% πιο γρήγορος και απο την συνάρτηση Quadprog του Matlab αλλά και απο το Cplex. Παρόλα αυτά, όταν έγινε η νέα παραμετρικοποίηση στο Quadprog και Cplex το GPAD ήταν κατά 96% γρηγορότερο ώς προς το Quadprog και περίπου 11% γρηγορότερο ώς προς το Cplex.

Στο πρόβλημα της αποστακτικής μονάδας επίσης ήταν πιο γρήγορος και από τους 2 άλλους κώδικες. Στο πρόβλημα δικής μας κατασκευής ο αλγόριθμος GPAD δεν απέδωσε ικανοποιητικά αλλά αυτό οφειλόταν στο ότι ο πίνακας που δημιουργήσαμε στον τετραγωνικό όρο ήταν μη ομαλός ειδικά για μεγάλους ορίζοντες κάτι που όμως είναι αναμενόμενο καθώς οι τιμές που χρησιμοποιήσαμε στο αρχικό σύστημα μεταβλητών κατάστασης ήταν αυθαίρετοι και τυχαίοι (το γεγονός αυτό φάνηκε και από το ότι ούτε η Quadprog έλυνε το πρόβλημα για σχετικά μεγάλους ορίζοντες).

Προτάσεις για έρευνα:

Από τα αποτελέσματα των πειραμάτων, , παρατηρούμε ότι υπάρχει μια εκλεκτικότητα των αλγορίθμων ανάλογα με το είδος του προβλήματος. Αυτό σημαίνει ότι ένας αλγόριθμος «Α» μπορεί να είναι πολύ πιο γρήγορος στην εύρεση της ακριβής λύσης σε ένα πρόβλημα X_1 σε σχέση με έναν αλγόριθμο «Β». .Δεν μπορούμε να ξέρουμε όμως ποιός από τους 2 θα είναι αποτελεσματικότερος για ένα άλλο πρόβλημα X_2. Είδαμε δηλαδή ότι κάποιοι αλγόριθμοι γίνονται από πολύ γρήγοροι όταν διαχειρίζονται μικρό αριθμό μεταβλητών κατάστασης και εισόδου (πχ αλγόριθμοι explicit MPC) ενώ γίνονται πολύ αργοί όταν αλλάζει το προς μελέτη σύστημα. Αντίστοιχα είδαμε ότι ο GPAD είναι πολύ πιο αργός από τον active-set αλγόριθμο του Quadprog όταν ο πίνακας στο τετραγωνικό μέρος δεν είναι ομαλός. Επίσης ήδη γίνεται έρευνα σχετικά με εναλλακτικές υπολογισμού της κλίσης της δυικής συνάρτησης καθώς αυτό το κομμάτι του αλγορίθμου είναι αυτό που φέρει το μεγαλύτερο υπολογιστικό φορτίο.

Έτσι λοιπόν υπάρχει ανάγκη για έρευνα στον τομέα των κριτηρίων που μπορούν να καθορίζουν ποιός τύπος αλγορίθμου θα είναι αποτελεσματικότερος στο κατά περίπτωση πρόβλημα. Η δομή του χώρου που δημιουργείται από τους περιορισμούς καθώς και το μέγεθός του, έχουν μεγάλη σημασία στον καθορισμό του αν ένας αλγόριθμος θα είναι αποδοτικός ή όχι. Συγκεκριμένα για την περίπτωση των τετραγωνικών προβλημάτων που συναντάμε σε εφαρμογές MPC, ήδη γίνεται έρευνα για την αξιοποίηση της μορφής-δόμης που έχουν οι σχηματιζόμενοι πίνακες στον ορίζοντα πρόβλεψης (βλέπε: Διαμόρφωση QP), ένα στοιχείο που αν αξιοποιηθεί σωστά μπορεί να μειώσει σε πολύ μεγάλο βαθμό τους χρόνους εκτέλεσης.

Βιβλιογραφία:

[1]. *Convex Optimization* Stephen Boyd and Lieven Vandenberghe

[2]. Bemporad, Patrinos, Simple and Certifiable Quadratic Programming Algorithms for Embedded Linear Model Predictive Control, IMT italy 2012

[3]. Borrelli et.al, Brief paper on the computation of linear model predictive control laws, 2010

[4]. Bemporad. Model-based predictive control design: New trends and tools. In Proc. 45th IEEE Conf. on Decision and Control , pages 6678{6683, San Diego, CA, 2006.

[5]. Nesterov. A method of solving a convex programming problem with convergence rate **O** (1**=k2**). Soviet Mathematics Doklady , 27(2):372{376, 1983.

[6]. Richter, M. Morari, and C.N. Jones. Towards computational complexity certification for constrained MPC based on Lagrange relaxation and the fast gradient method. In Proc. 50th IEEE Conf. on Decision and Control and European Control Conf. , pages 5223{5229, Orlando, USA, 2011.

[7]. Δάλλα, Κυρτά Σύνολα και Εφαρμογές 1999

[8]. Γ. Σίσκος, Γραμμικός Προγραμματισμός

[9]. Nesterov. Introductory lectures on convex optimization: A basic course Kluwer Academic Publishers, 2004.

[10]. http://www.kuleuven.be/optec/software/onlineQP?start=1

[11]. https://en.wikipedia.org/wiki/Simplex

[12]. Wong, Active-set methods for quadratic programming, Doctor in mathematics thesis

[13]. Μαυρωτάς, Δυϊκή θεωρία, Διαφάνειες στο μάθημα Επιχειρησιακή Έρευνα Σχολή Χημικών Μηχανικών 4ο εξάμηνο

[14]. Κυρανούδης, Μηχανική Συστημάτων Εφοδιαστικής Διαχείρισης,2005

[15]. www.mathworks.com

ΠΑΡΑΡΤΗΜΑ: Υλοποίηση αλγόριθμου GPAD στο MATLAB

```
X0=x_opt(1,:);
I=eye(size(H,1));
HQ=H;
fQ=g(1,:);
AQ=[A;-A];
bQ=[ubA(1,:)';-lbA(1,:)'];
t=tic;%Sets timer to Time the run of quadprog algorithm!
[X,fval]=quadprog(HQ,fQ,AQ,bQ,[],[],lb(1,:)',ub(1,:)');
elapsedquadprog=toc(t);%Records time to run quadprog function

%% we have to make sure H_cplex matrix is exactly symmetric so that cplex
won't crash.
for i=1:size(HQ,1)
    for j=i+1:size(HQ,1)
        HQ(i,j)=HQ(j,i);
    end
end
%% CPLEX QP PERFORMANCE:
try
    t=tic;
    [cplex_solutin,fval_cplex]=cplexqp(HQ,fQ',AQ,bQ,[],[],lb(1,:)',ub(1,:)');
    elapsed_cplex=toc(t);%Records time to run quadprog function
catch
 disp('an error occured while trying to run Cplex algorithm....check if u
have 32 bit of matlab or cplex algos installed')
    elapsed_cplex='zero due to error occured while trying to run the cplex
algorithm';
end
%% GPAD algorythm-Declaration of Matrices
%Checking time and results which will be displayed plus given duality gap

X0=x_opt(1,:);
I=eye(size(H,1));
HQ=H;%M Matrix

fQ=g(1,:);%this is noted as (Cp+g)' in our paper
AQ=[A;-A;I;-I];%this is noted as G in our paper
%bQ=[lbA(1,:)';-ubA(1,:)';lb(1,:)';-ub(1,:)'];%this is noted as Ep-b in our
paper
bQ=[ubA(1,:)';-lbA(1,:)';ub(1,:)';-lb(1,:)'];
y=zeros(size(bQ,1),1);
y_minus1=zeros(size(y,1),1);
z_v=zeros(nV,1);
Counter=0;

MINV=inv(HQ);

g_p=MINV*fQ';
M_g=MINV*AQ';
L=max(eig(HQ));
G_L=AQ/L;
p_D=-(bQ)/L;
theta=[1,1];%as defined in(8e)
beta=theta(2)*(theta(1).^(-1)-1);%theta(2)is the most recent whereas
theta(1)is one iteration before theta(1)
% theta(1)=theta(2);
% theta(2)=(sqrt(theta(1)^4+4*theta(1)^2)-theta(1)^2)/2;
%V=Dualfi(HQ,MINV,fQ,AQ,bQ,y,z_v);
z_vminus1=z_v;

%% Calculations which will be used to find the sum of Dual and Primal
```

```
function
D_mod=AQ*MINV*fQ'+bQ;%Dp +d matrix component
D_mod=D_mod';
d_p=(1/2)*fQ*MINV*fQ';
H_moded=AQ*MINV*AQ';


V=Dualfotos(HQ,H_moded,d_p,D_mod,fQ,y,z_v);

%% FANCY INPUT WINDOWS!
prompt = {'Enter Duality Gap (Tolerance) in (0,1):'};
dlg_title = 'Input';
num_lines = 1;
def = {'10^(-8)'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
R=cell2mat(answer);
R=str2num(R);

prompt = {'DO YOU WISH TO GET PLOTS ABOUT GPADs PERFORMANCE?'};
dlg_title = 'PERFORMANCE CHECK';
Check_to_iterate= questdlg(prompt,dlg_title);
if (R<0||R>1)&&strcmp(Check_to_iterate,'Yes')
    R=10^(-5);
end


if strcmp(Check_to_iterate,'Yes')&&R<1&&R>0
    End_of_iterations=-log10(R);
    m=1;%will be used to record the values of counter,time, etc
    step=0.1;
else
    End_of_iterations=1;
    step=1;
    m=1;
end
%%Dimensions for recording data on algorithms performance
eg_vector=zeros(size(1:step:End_of_iterations,2),1);
elapsed_vector=eg_vector;
Counter_vector=eg_vector;
errorvector_tracker=eg_vector;


for i=1:step:End_of_iterations
    if strcmp(Check_to_iterate,'Yes')
        eg=10.^(-i);
    eg_vector(m)=eg;
    else
        eg=R;
    end
    %% I CONNECT THIS PART WITH THE QUADPROG FORMULATION
tonos=tic;%Sets the Timer to calculate time it takes for the GPAD algorithm
to run


%% Application:(specify eg-tolerance- evgap)
ev=eg;
while (sum(gt(AQ*(-z_v)-bQ,eg)))~=0||(gt(V,ev))~=0
    %step1:find w,z,y(counter+1)

    w_v=y+beta*(y-y_minus1);
    z_app=-M_g*w_v-g_p;%approximation of primal
    if Counter>0
        y_minus1=y;
    end
    y=(w_v+G_L*z_app+p_D);%Dual solution

    help_affine_y=(y<0);%variable that helps make all <0 elements
    %equal to 0
    y(help_affine_y)=0;%all negative values become zero
```

```
    V=Dualfotos(HQ,H_moded,d_p,D_mod,fQ,y,z_v);
    %V=Dualfi(HQ,MINV,fQ,AQ,bQ,y,z_v);
    %  Calculation of terms that define Z_v(our primal solution)
    if Counter>0
     z_v=(1-theta(2))*z_v-theta(2)*z_app;%Primal solution
    end

    %Calculations needed for the next iteration:
    theta(1)=theta(2);
    theta(2)=(sqrt(theta(2)^4+4*theta(2)^2)-theta(2)^2)/2;

    assert(theta(2)~=theta(1),'SAME THETAS')%CONSISTENCY CHECK
    beta=theta(2)*(theta(1).^(-1)-1);
    y_minus1=y;
    %Break-Loop-Condition
    Counter=Counter+1;
%     if Counter>500
%         disp('Note: Counter exceeded 500 iterations')
%         break;
%     end
    Counter_vector(m)=Counter;
    Counter=Counter+1;


end
elapsed=toc(tonos); %Time GPAD algorithm needs
elapsed_vector(m)=elapsed;
errorvector=-z_v-X0';%calculating errors for given dual gap
errorvector_tracker(m)=max(abs(errorvector));

m=m+1;


end

if strcmp(Check_to_iterate,'Yes')&&m~=2%check for plotting

    subplot(3,1,1)
    loglog(eg_vector,elapsed_vector,'r+:')
    xlabel('Duality gap tolerance')
    ylabel('Time in seconds')
    subplot(3,1,2)
    loglog(eg_vector,Counter_vector)
    xlabel('Tolerance')
    ylabel('N. of Iterations')
    subplot(3,1,3)
    loglog(eg_vector,errorvector_tracker,'mx')
    xlabel('Tolerance')
    ylabel('Error (maximum) ')

end
disp('The optimal vector is : ')
optimum_GPAD=-z_v;
errorvector=((optimum_GPAD-X0')./X0')*100;%calculating the percentage
difference of errors
disp(' The maximum error is at position : ')
disp(find(abs(errorvector)==max(abs(errorvector))))
disp('of error_vector and has the value of : ')
disp(max(abs(errorvector)))
disp('The value of the primal function for the optimal vector is : ')
value_of_primal_function=Dualfotos(HQ,zeros(size(H_moded)),zeros(size(d_p)),z
eros(size(D_mod)),fQ,zeros(size(y)),z_v)
V=Dualfotos(HQ,H_moded,d_p,D_mod,fQ,y,z_v);


disp('Quadprog Matlab function took');disp(elapsedquadprog)
disp('seconds')
disp('Whereas the GPAD algorithm took:')
disp(elapsed)
disp('seconds')
```

```
disp('for a duality gap=');eg
if (R<0||R>1)&&strcmp(Check_to_iterate,'Yes')
    warning('Because Duality Gap has to be in (0,1), and you asked for plots,
Matlab run for the Default Gap')
end
msgbox(cat(2, {'Optimal value of primal function
(V):'},num2str(value_of_primal_function),{' '},{'Quadprog Matlab function
took: '},[num2str(elapsedquadprog) ' seconds'],{' '},{'Cplex_qp function took
: '},[num2str(elapsed_cplex) ' seconds'],{' '},{'GPAD Algorithm took (for the
given Dual Gap) :'},[num2str(elapsed) ' seconds'],{' '},{['Error in solution
for an error gap = ' num2str(R)]}, [num2str(max(abs(errorvector)))
'%']),'Results:')

%FUNCTION for calculation of sum of primal and dual:

function V=Dualfotos(HQ,H_moded,d_p,D_mod,fQ,y,z_v)
%takes input vector y size(size(bQ,1),1) and the defining matrices
%calculates output of dual equivalent
%Reminder:HQ<->M,fQ<->(Cp+g)',G<->AQ,

z_v=-z_v;
V=(1/2)*(y'*H_moded*y+z_v'*HQ*z_v)+fQ*z_v+D_mod*y+d_p;
end
```