Προσομοίωση κυκλωμάτων μεγάλης κλίμακας με τεχνικές θεωρίας γράφων

Νόνας Ευάγγελος

Διπλωματική εργασία για την απόκτηση
του Διπλώματος του Μηχανικού Ηλεκτρονικών Υπολογιστών,
Τηλεπικοινωνιών και Δικτύων

του τμήματος

Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Πανεπιστημίου Θεσσαλίας

Επιβλέποντες καθηγητές:

Ευμορφόπουλος Νέστωρ, Επίκουρος Καθηγητής
Σταμούλης Γεώργιος, Καθηγητής

Σεπτέμβριος 2015

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ

Προσομοίωση κυκλωμάτων μεγάλης κλίμακας με τεχνικές θεωρίας γράφων

Διπλωματική Εργασία

Νόνας Ευάγγελος

Επιβλέποντες καθηγητές:

Ευμορφόπουλος Νέστωρ, Επίκουρος Καθηγητής
Σταμούλης Γεώργιος, Καθηγητής

Εγκρίθηκε από διμελή εξεταστική επιτροπή την ..... Σεπτεμβρίου 2015

.....	.....
Ν.Ευμορφόπουλος	Γ.Σταμούλης
Επίκουρος Καθηγητής	Καθηγητής

Διπλωματική εργασία για την απόκτηση του Διπλώματος του Μηχανικού Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας, στα πλαίσια του Προγράμματος Προπτυχιακών Σπουδών του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Θεσσαλίας.

.....

Νόνας Ευάγγελος
Διπλωματούχος Μηχανικός Υπολογιστών, Τηλεπικοινωνιών και Δικτύων Παπιστημίου Θεσσαλίας

Στην οικογένεια και στους φίλους μου.

Θα ήθελα να ευχαριστήσω θερμά όλους όσους συνέβαλαν στο να περατωθεί η παρούσα διπλωματική εργασία και πρωτίστως, τους επιβλέποντες καθηγητές μου κ.Νέστορα Ευμορφόπουλο και κ.Γεώργιο Σταμούλη για την συνεχή καθοδήγηση τους καθ'όλη τη διάρκεια της εργασίας.

Στη συνέχεια θα ήθελα να ευχαριστήσω τους συναδέλφους στο γραφείο Ε5 και ιδιαίτερα τον κ.Χαράλαμπο Αντωνιάδη. Να μην παραλείψω βέβαια το μεγάλο ευχαριστώ που οφείλω στον κ.Κωνσταντή Νταλούκα για την πολύτιμη βοήθεια του σε θέματα αποσφαλμάτωσης του κώδικα.

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου και την οικογένεια μου, που ήταν πάντα στο πλευρό μου καθ'όλη τη διάρκεια της φοίτησής μου.

Νόνας Ευάγγελος,
Βόλος 2015

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Το πρόβλημα της προσομοίωσης κυκλωμάτων

Το πρόβλημα της προσομοίωσης κυκλωμάτων συνίσταται στην χρήση λογισμικού το οποίο μέσω μαθηματικών μοντέλων αναπαράγει την συμπεριφορά ενός πραγματικού ηλεκτρικού κυκλώματος.

Στην σύγχρονη εποχή το μέγεθος και η πολυπλοκότητα των ηλεκτρικών κυκλωμάτων καθιστά αδύνατη τη μελέτη της συμπεριφοράς ενός ηλεκτρικού κυκλώματος χωρίς την χρήση του απαραίτητου λογισμικού (προσομοιωτή), το οποίο θα εκτελεστεί σε ένα υπολογιστικό σύστημα.

Οι προσομοιωτές αναλαμβάνουν να υπολογίσουν τις τάσης των κόμβων του κυκλώματος, καθώς και τα ρεύματα ορισμένων κλάδων. Μέσω της τροποποιημένης ανάλυσης κόμβων (*modified nodal analysis*), της πιο διαδεδομένης μεθόδου μοντελοποίησης ηλεκτρικών κυκλωμάτων, ο προσομοιωτής δημιουργεί εξισώσεις, η λύση των οποίων μας δίνει τα επιθυμητά αποτελέσματα.

Η επίλυση των εξισώσεων αυτών έγκειται εν τέλει στην επίλυση ενός γραμμικού συστήματος. Ωστόσο, καθώς η κλίμακα αυξάνεται όλο και περισσότερο, τα ολοκληρωμένα κυκλώματα (*integrated circuits*), καθώς και τα δίκτυα παροχής ηλεκτρικής ενέργειας (*power delivery networks*), έχουν γίνει τόσο περίπλοκα, που ακόμα και οι προσομοιωτές απαιτούν αρκετούς υπολογιστικούς πόρους και αρκετό χρόνο για την επίλυση των γραμμικών συστημάτων τα οποία μας δίνουν τα αποτελέσματα της προσομοίωσης.

Για τον λόγο αυτό οδηγούμαστε στην αναζήτηση τεχνικών για την βελτίωση των προσομοιωτών, ώστε να επιτυγχάνεται ταχύτερη επίλυση των γραμμικών συστημάτων. Πέρα από την χρήση υπολογιστικά ισχυρότερων μηχανημάτων, μεγάλο ερευνητικό ενδιαφέρον στρέφεται προς την κατεύθυνση αναζήτησης ταχύτερων αλγορίθμων για την επίλυση γραμμικών συστημάτων.

1.2 Η συμβολή της παρούσας εργασίας

Στην παρούσα διπλωματική εργασία θα εφαρμόσουμε τεχνικές από την σύγχρονη θεωρία γράφων και την θεωρία στήριξης ($support-theory$), για την εξαγωγή αποδοτικών προρυθμιστών κατάστασης (*preconditioners*), ώστε να επιτυχυνθεί η ταχύτητα σύγκλισης της μεθόδου συζυγών κλίσεων (*conjugate gradient*).

Στόχος είναι να αναλύσουμε τους αλγορίθμους για την κατασκευή των προρυθμιστών κατάστασης και να συγκρίνουμε το κόστος κατασκευής, το κόστος παραγοντοποίησής, και τέλος την απο-

δοτικότητα τους στην επίλυση των γραμμικών συστημάτων. Θα δούμε πόσο μπορεί να βελτιωθεί ο αριθμός επαναλήψεων της *conjugate gradient* και αντίστοιχα ο χρόνος εύρεσης της λύσης.

Οι προρυθμιστές κατάστασης βασίζονται σε υπογράφους του γράφου του κυκλώματος και πιο συγκεκριμένα το ελάχιστο επικαλύπτον δέντρο, το επικαλύπτον δέντρο χαμηλής έκτασης και το επαυξημένο επικαλύπτον δέντρο χαμηλής έκτασης.

1.3 Διάρθρωση της εργασίας

Στο κεφάλαιο 2 θα παρουσιάσουμε όλες τις βασικές έννοιες που αφορούν τα γραμμικά ηλεκτρικά κυκλώματα, καθώς και οι απαραίτητες αρχές στις οποίες βασίζεται η μοντελοποίηση αυτών. Θα μελετήσουμε τα γραμμικά κυκλωματικά στοιχεία και τις χαρακτηριστικές εξισώσεις τους. Στην συνέχεια, θα δούμε πως μπορούμε, μέσω των νόμων του $Kirchhoff$ και της τροποποιημένης ανάλυσης κόμβων να μοντελοποίησουμε το κύκλωμα, εξάγωντας τις εξισώσεις που διέπουν τη συμπεριφορά του. Μετά την μοντελοποίηση του κυκλώματος, η προσομοίωση του έγκυται στην επίλυση γραμμικών συστημάτων.

Στο κεφάλαιο 3 θα μελετήσουμε αλγόριθμους για την επίλυση γραμμικών συστημάτων. Παρατίθενται οι άμεσες μέθοδοι επίλυσης συστημάτων, μέσω της παραγοντοποίησης LU ή της παραγοντοποίησης $Cholesky$, καθώς και οι επαναληπτικές μέθοδοι *conjugate gradient* και $bi-conjugate\ gradient$. Ιδιαίτερη έμφαση δίνεται στην ανάλυση της *conjugate gradient* και στην έννοια των προρυθμιστών κατάστασης *preconditioners*). Μέσω αποδοτικών προρυθμιστών κατάστασης θα δείξουμε πως μπορεί να επιταχυνθεί σημαντικά η σύγκλιση της μεθόδου, οδηγώντας έτσι σε γρηγορότερη και αποδοτικότερη επίλυση συστημάτων.

Στο κεφάλαιο 4 θα παρουσιάσουμε αλγορίθμους που προκύπτουν από την θεωρία γράφων για την κατασκευή αποδοτικών προρυρθμιστών κατάστασης. Θα αναλύσουμε τον ισομορφισμό μεταξύ Λαπλασιανών πινάκων και βεβαρημένων γράφων και θα δούμε την διαδικασία εξαγωγής προρυθμιστών κατάστασης μέσω των αλγορίθμων αφορούν την κατασκευή ελάχιστου επικαλύπτοντος δέντρου, επικαλύπτοντος δέντρου χαμηλής έκτασης και επαυξημένου επικαλύτοντος . Επίσης, θα παρουσιάσουμε συνοπτικά το αναγκαίο θεωρητικό υπόβαθρο όπως αυτό προκύπτει από την θεωρία γράφων και την σύγχρονη θεωρία στήριξης (*support theory*).

Στο κεφάλαιο 5 θα δούμε πως μπορούμε να επιτύγχουμε ακομά ταχύτερους επιλυτές μέσω της αναδρομικής επίλυσης συστημάτων των προρυθμιστών κατάστασης μέσω αναδρομικού *preconditioning*. Θα παρουσιάσουμε την διαδικασία της μερικής παραγοντοποίησης $Cholesky$ (*partial Cholesky factorization*) καθώς και τους αλγορίθμους με τους οποίους επιτυγχάνεται το αναδρομικό *preconditioning* και η αντίστοιχη αναδρομική επίλυση των συστημάτων που θα προκύψουν.

Τέλος, στο κεφάλαιο 6 θα αναλύσουμε την πειραματική διαδικασία εξαγωγής των αποτελεσμάτων. Αφού παρουσιάσουμε τα πειράματα θα εξετάσουμε και θα ερμηνεύσουμε τα αποτελέσματα, προκειμένου να καταλήξουμε σε συμπεράσματα σχετικά με την αποδοτικότητα, την αποτελεσματικότητα και το κόστος κατασκευής του εκάστοτε αλγορίθμου. Επίσης, θα δούμε, περιληπτικά, μελλοντικές επεκτάσης της παρούσας εργασίας.

Κεφάλαιο 2

Μοντελοποίηση ηλεκτρικών κυκλωμάτων

2.1 Βασικές έννοιες ηλεκτρικών κυκλωμάτων

Ένα ηλεκτρικό κύκλωμα είναι ένα διασυνδεδεμένο δίκτυο ηλεκτρικών στοιχείων στο οποίο δημιουργούνται κλειστές, αγώγιμες από το ηλεκτρικό ρεύμα διαδρομές. Στην παρούσα εργασία ασχολούμαστε με γραμμικά κυκλώματα, τα οποια υπακούν την αρχή της επαλληλίας (*superposition principle*). Έτσι, η έξοδος ενός κυκλώματος $F(x)$, όταν εφαρμόζεται ένας γραμμικός συνδυασμός σημάτων $\alpha_1 x_1(t) + \alpha_2 x_2(t)$ ως είσοδος, θα ισούται με τον γραμμικό συνδυασμό των εξόδων των σημάτων $x_1(t)$ και $x_2(t)$ όταν αυτά εφαρμόζονται ξεχωριστά, δηλαδή:

$$F\{\alpha_1 x_1(t) + \alpha_2 x_2(t)\} = \alpha_1 F\{x_1(t)\} + \alpha_2 F\{x_2(t)\}.$$

Άτυπα, μπορούμε να θεωρούμε γραμμικό, ένα ηλεκτρικό κύκλωμα του οποίου οι τιμές των ηλεκτρικών στοιχείων (αντίσταση, χωρητικότητα, επαγωγή, κέρδος, κτλ.) δεν αλλάζουν σε σχέση με τα επίπεδα τάσης και ρεύματος του κυκλώματος.

Στα ηλεκτρικά κυκλώματα διακρίνουμε τα εξής μέρη:

- Κόμβος (*node*): είναι ένα σημείο αναφοράς του κυκλώματος, στο οποίο συντρέχουν περισσότερα από δύο κυκλωματκά στοιχεία. Συνήθως, οι κόμβοι επισημαίνονται με ονόματα ως σημεία αναφοράς για την μελέτη του κυκλώματος.
- Κλάδος (*branch*): είναι το μέρος του κυκλώματος μεταξύ δύο διαδοχικών κόμβων.
- Βρόγχος (*loop*): είναι ένα μερος ενός κυκλώματος που ορίζεται από διαδοχικούς κλάδους, σχηματίζοντας μια κλειστή διαδρομή.

Για την περιγραφή και την ανάλυση των ηλεκτρικών κυκλωμάτων υπάρχουν νόμοι, οι οποίοι ισχύουν για όλα τα ηλεκτρικά κυκλώματα:

- Οι νόμοι του $Kirchhoff$ ($Kirchhoff's\ circuit\ laws$)
- Ο (μακροσκοπικός) νόμος του Ohm ($Ohm's\ law$)
- Τα θεωρήματα $Thevenin$ και $Norton$ ($Thevenin's\ theorem,\ Norton's\ theorem$)
- Το θεώρημα της επαλληλίας (*Superposition principle*)

Οι νόμοι του $Kirchhoff$ αφορούν τους κόμβους και τους βρόγχους του κυκλώματος και αναλύονται εκτενέστερα σε επόμενη παράγραφο. Σε αυτούς βασίζεται η μαθηματική μοντελοποίηση του κυκλώματος για την προσομοίωσή του. Ο νόμος του Ohm συσχετίζει την τάση με το ρεύμα σε ένα μεμονωμένο κλάδο του κυκλώματος. Στην επόμενη παράγραφο, χρησιμοποιούμε τον νόμο του Ohm για τον υπολογισμό της σχέσης τάσης-ρεύματος του αντιστάτη. Παρακάτω, αναφέρουμε τα θεωρήματα $Thevenin$ και $Norton$, καθώς και το θεώρημα της επαλληλίας. Παρ΄όλο που δεν τα χρησιμοποιούμε στην παρούσα εργασία τα παραθέτουμε για λόγους πληρότητας, αλλά και της σπουδαιότητας αυτών. Το θεώρημα $Thevenin$ μας λέει πως: Όποιοδήποτε κύκλωμα που αποτελείται από πηγές τάσης ή ρεύματος και αντιστάσεις είναι ηλεκτρικά ισοδύναμο με μία μοναδική πηγή τάσης, σε σειρά με μία μοναδική αντίσταση΄. Αντίστοιχα, το θεώρημα $Norton$ μας λέει πως: Όποιοδήποτε κύκλωμα που αποτελείται από πηγές τάσης ή ρεύματος και αντιστάσεις είναι ηλεκτρικά ισοδύναμο με μία μοναδική ιδανική πηγή ρεύματος, παράλληλα με μία μοναδική αντίσταση΄. Τα ισοδύναμα κυκλώματα $Thevenin$ και $Norton$ συνδέονται μεταξύ τους μέσω των σχέσεων:

$$R_{TH} = R_{NO},$$

$$V_{TH} = I_{NO} R_{NO},$$

$$I_{NO} = \frac{V_{TH}}{R_{TH}}.$$

Τέλος, το θεώρημα της επαλληλίας μας λέει πως σ΄ένα γραμμικό κύκλωμα με πολλές ανεξάρτητες πηγές, η απόκριση σ΄ένα συγκεκριμένο κλάδο όταν όλες οι πηγές δρουν ταυτόχρονα ισούται με το γραμμικό άθροισμα των ατομικών αποκρίσεων της κάθε πηγής/διέγερσης ξεχωριστά.

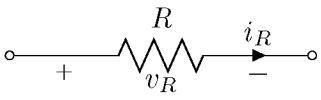
2.2 Βασικά κυκλωματικά στοιχεία

Διακρίνουμε τα κυκλωματικά στοιχεία σε δύο μεγάλες κατηγορίες: τα ενεργά και τα παθητικά στοιχεία.

2.2.1 Παθητικά στοιχεία

Τα παθητικά στοιχεία είναι εκείνα τα κυκλωματικά στοιχεία τα οποία καταναλώνουν (αλλά δεν παράγουν) ενέργεια. Η ενέργεια που καταναλίσκεται είτε αποθηκεύεται (σε ενέργεια ηλεκτρικού ή μαγνητικού πεδίου) είτε μετατρέπεται σε άλλη μορφή ενέργειας (π.χ. θερμική ενέργεια), χωρίς βέβαια να ενισχύεται η ισχύς της εξόδου των στοιχείων αυτών. Τα κυριότερα παθητικά στοιχεία είναι τα ακόλουθα:

1. Αντίστατες: Ο αντιστάτης ανήκει στην κατηγορία παθητικών στοιχείων που μετατρέπουν την ηλεκρτρική ενέργεια σε θερμική. Ο αντιστάτης συνδέεται με την έννοια της αντίστασης, δηλαδή την δυσκολία (αντίσταση) διέλευσης ηλεκτρικού ρεύματος δια μέσου ενός αγωγού, δεδομένης διαφοράς δυναμικού στα άκρα του αγωγού (στην φυσική παραλληλίζεται -είναι το ηλεκτικό ανάλογο- με την έννοια της μηχανικής τριβής). Μονάδα μέτρησης της αντίστασης στο διεθνές σύστημα μονάδων ($S.I.$) το Ohm (Ω). Η αντίθετη έννοια, ονομάζεται ηλεκτρική αγωγιμότητα και αφορά την ευκολία διέλευσης ηλεκτρικού ρεύματος δια μέσω ενός αγωγού δεδομένης διαφοράς δυναμικού στα άκρα του. Μονάδα μέτρησης της ηλεκτρικής αγωγιμότητας



στο διεθνές σύστημα μονάδων $(S.I.)$ το $Siemens$ (S). Η σχέση που συνδέει την ηλεκτρική αντίσταση (r) μη την ηλεκτρική αγωγιμότητα (g) είναι η ακόλουθη:

$$g = \frac{1}{r} \Leftrightarrow r = \frac{1}{g}.$$

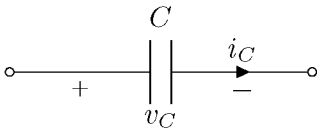
Ο νόμος του Ohm μας λέει οτι σ' έναν αντιστάτη αντίστασης r (και αγωγιμότητας g) στα άκρα του οποίου εφαρμόζεται τάση $v(t)$ και διαρρέεται από ρεύμα έντασης $i(t)$, η ένταση του ρεύματος είναι ανάλογη της τάσης με συντελεστή αναλογίας $\frac{1}{r} = g$. Έτσι λοιπόν ένας αντιστάτης χαρακτηρίζεται από την εξίσωση:

$$i(t) = \frac{1}{r}v(t) = gv(t).$$

2. Πυκνωτές: Ο πυκνωτής καταναλώνει ενέργεια την οποία αποθηκεύει σε μορφή ηλεκτρικού πεδίου. Αποτελείται από ένα σύστημα γειτονικών αγωγών (οπλισμοί) αναάμεσα στους οποίους παρεμβάλλεται μονωτικό υλικό (διηλεκτρικό). Όταν ο πυκνωτής φορτίζεται οι οπλισμοί του αποκτούν ίσα και αντίθετα φορτία ($+q$ και $-q$ αντίστοιχα) και μεταξύ αυτών αναπτύσσεται διαφορά δυναμικού v. Ο πυκνωτής συνδέεται με την έννοια της χωρητικότητας. Η χωρητικότητα ορίζεται ως το πηλίκο του φορτίου προς την τάση του πυκνωτή

$$c = \frac{q}{v} \equiv \frac{q(t)}{v(t)},$$

και σχετίζεται με τα γεωμετρικά του χαρακτηριστικά και την φύση του διηλεκτρικού. Μονάδα μέτρησης της χωρητικότητας στο διεθνές σύστημα μονάδων $(S.I.)$ είναι το $Farad$ (F).



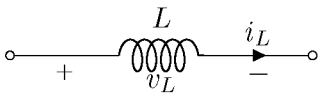
Χρησιμοποιώντας την σχέση της χωρητικότητας μπορούμε να εξάγουμε την χαρακτηριστική εξίσωση του πυκνωτή:

$$c = \frac{q(t)}{v(t)} \Leftrightarrow q(t) = cv(t) \Leftrightarrow \frac{dq(t)}{dt} = c\frac{dv(t)}{dt} \Leftrightarrow i(t) = c\frac{dv(t)}{dt}.$$

3. Πηνία: Το πηνίο καταναλώνει ενέργεια, την οποία αποθηκεύει σε μορφή μαγνητικού πεδίου, καθώς έχει την ιδιότητα να αναπτύσσει μαγνητικό πεδίο όταν διαρρέεται από ρεύμα. Το πηνίο συνδέεται με την έννοια της αυτεπαγωγής, άμεσο επακόλουθο της ηλεκτρομαγνητικής επαγωγής και του κανόνα του $Lenz$:

$$\varepsilon = -\frac{\partial \Phi_B}{\partial t}.$$

Μονάδα μέτρησης της αυτεπαγωγής στο διεθνές σύστημα μονάδων $(S.I.)$ είναι το $Henry$ (H).



Προκύπτει λοιπόν οτι για ένα πηνίο με συντελεστή αυτεπαγωγής l, το οποίο διαρρέεται από ρεύμα έντασης $i \equiv i(t)$, η τάση αυτεπαγωγής που δημιουργείται στα άκρα του θα είναι:

$$v = l\frac{di}{dt},$$

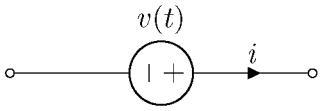
και άρα σ΄ έναν αντίστοιχο κλάδο του κυκλώματος θα ισχύει:

$$v(t) = l\frac{di(t)}{dt}.$$

2.2.2 Ενεργά στοιχεία

Ένα κυκλωματικό στοιχείο το οποίο δεν είναι παθητικό καλείτε ενεργό και παρέχει ηλεκτρική ενέργεια στο κύκλωμα. Τα ενεργά στοιχεία αναφέρονται συχνά και ως πηγές. Στην εργασία αυτή μας ενδιαφέρουν οι ανεξάρτητες πηγές τάσης και οι ανεξάρτητες πηγές ρεύματος.

1. Ανεξάρτητες πηγές τάσης: Πηγή τάσης είναι ένα κυκλωματικό στοιχείο το οποίο παρέχει ηλεκτρική ενέργεια στο κύκλωμα μέσω την προσφοράς τάσης (διαφοράς δυναμικού). Η ανεξάρτητη πηγή τάσης έχει διαφορά δυναμικού στα άκρα της ανεξάρτητη από το ρεύμα που την διαρρέει.

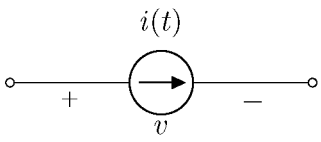


Η εξίσωση που την περιγράφει είναι η ακόλουθη:

$$v(t) = a + f(t) \equiv s(t).$$

Παρατηρούμε πως η εξίσωση είναι ανεξάρτητη την ένταση του ρεύματος ($i(t)$) και πως εμπεριέχει μία dc συνιστώσα (α) και μία $transient$ συνιστώσα ($f(t)$).

2. Ανεξάρτητες πηγές ρεύματος: Πηγή ρεύματος είναι ένα κυκλωματικό στοιχείο το οποίο παρέχει ηλεκτρική ενέργεια στο κύκλωμα μέσω την προσφοράς ρεύματος. Η ανεξάρτητη πηγή ρεύματος προσφέρει ρεύμα στο κύκλωμα, του οποίου η ένταση είναι ανεξάρτητη από την τάση στα άκρα της πηγής.



Η εξίσωση που την περιγράφει είναι:

$$i(t) = a + f(t) \equiv s(t).$$

Παρατηρούμε πως η εξίσωση είναι ανεξάρτητη της τάσης ($v(t)$) και πως εμπεριέχει μία dc συνιστώσα (α) και μία $transient$ συνιστώσα ($f(t)$)

Για όλα τα κυκλωματικά στοιχεία ακολουθούμε την συζευγμένη φορά τάσης ρεύματος: το ρεύμα εισέρχεται απο τον ακροδέκτη υψυλότερου δυναμικού (+) και εξέρχεται από τον ακροδέκτη χαμηλότερου δυναμικού (-). Έτσι σε κάθε κλάδο του κυκλώματος θεωρούμε θετικό το ρεύμα που ακολουθεί την συζευγμένη φορά, ενώ στην αντίθετη περίπτωση τον ρεύμα θεωρείται αρνητικό.

2.3 Νόμοι του $Kirchhoff$

Οι νόμοι του $Kirchhoff$ εκφράζουν θεμελιώδεις αρχές διατήρησης της Φυσικής στα ηλεκτρικά κυκλώματα. Μπορούν να θεωρηθούν ως πορίσματα των εξισώσεων $Maxwell$ στο όριο των χαμηλών συχνοτήτων. Είναι απόλυτα ακριβείς στο DC, αλλά και στο AC, σε συχνότητες όμως όπου το μήκος κύματος της ηλεκτρομαγνητικής ακτινοβολίας είναι πολύ μεγάλο σε σχέση με το κύκλωμα. Οι νόμοι του $Kirchhoff$ είναι δύο: ο νόμος ρευμάτων του $Kirchhoff$ ($Kirchhoff's\ current\ law - KCL-$) και ο νόμος τάσεων του $Kirchhoff$ ($Kirchhoff's\ voltage\ law\ - KVL-$).

1. Ο νόμος ρευμάτων του $Kirchhoff$ μας λέει οτι: 'το αλγεβρικό άθροισμα των ρευμάτων σε κάθε κόμβο ενός κυκλώματος ισούται με το μηδέν'. Κατά σύμβαση, θεωρούμε θετικά τα ρεύματα τα οποία εισέρχονται στον κόμβο και αρνητικά αυτά τα οποία εξέρχονται. Επομένως, έστω οτι n είναι ο συνολικός αριθμός κλάδων ($branches$) με ρεύματα τα οποία ρέουν είτε προς, είτε από τον κόμβο, τότε έχουμε:
$$\sum_{k=1}^{n} i_k(t) = 0.$$
Μια ισοδύναμη διατύπωση είναι η εξής: 'σε κάθε κόμβο ενός κυκλώματος το άθροισμα των ρευμάτων που εισέρχονται σ' έναν κόμβο ισούται με το άθροισμα των ρευμάτων που εξέρχονται από αυτόν'. Ο νόμος ρευμάτων του $Kirchhoff$ είναι απόρροια της αρχής διατήρησης φορτίου.

2. Ο νόμος τάσεων του $Kirchhoff$ μας λέει οτι: 'Σε κάθε κλειστό βρόγχο ενός κυκλώματος, το άθροισμα των τάσεων (διαφορών δυναμικού) των επιμέρους κλάδων που απαρτίζουν τον βρόγχο ισούται με το μηδέν'. Παρόμοια λοιπόν με τον KCL διατυπώνεται μαθηματικά και ο KVL. Αν θεωρήσουμε οτι n είναι ο συνολικός αριθμός κλάδων ($branches$) που απαρτίζουν τον βρόγχο, τότε έχουμε:
$$\sum_{k=1}^{n} v_k(t) = 0.$$

2.4 Πίνακας πρόσπτωσης - Γενίκευση νόμων $Kirchhoff$

Από την θεωρία γράφων γνωρίζουμε οτι για ένα γράφο $G = (V, E)$ (όπου V το σύνολο κορυφών του και E το σύνολο των ακμών του) ορίζεται ο πίνακας πρόσπτωσης ($incidence\ matrix$). Ο πίνακας πρόσπτωσης έχει διάσταση $|V| \times |E|$ εφόσον οι γραμμές του αριθμούνται βάση των κορυφών και οι στήλες βάση των ακμών του γράφου. Για μην κατευθυνόμενο γράφο ο πίνακας πρόσπτωσης A, ορίζεται ώς:
$$\alpha_{ij} = \begin{cases} 1, \text{ αν η ακμή } j \text{ συνδέεται με την κορυφή } i \\ 0, \text{ διαφορετικά} \end{cases}$$

Ενώ αντίστοιχα, για κατευθυνόμενο γράφο, ο πίνακας πρόσπτωσης A, ορίζεται ως:

$$\alpha_{ij} = \begin{cases} 1, \text{ αν η ακμή } j \text{ εξέρχεται από την κορυφή } i \\ -1, \text{ αν η ακμή } j \text{ εξέρχεται από την κορυφή } i \\ 0, \text{ αν η ακμή } j \text{ δεν συνδέεται με την κορυφή } i \end{cases}$$

Επίσης, θεωρώντας ένα συνδεδεμένο γράφο $G = (V, E)$ με n κορυφές ($|V| = n$) και m ακμές ($|E| = m$) και A_α τον πίνακα πρόσπτωσης αυτού, τότε ο πίνακας A που προκύπτει διαγράφοντας μία οποισδήποτε γραμμή του πίνακα A_α, καλείται ελαττωμένος πίνακας πρόσπτωσης (*reduced incidence matrix*) του γράφου G.

Έτσι λοιπόν, για οποιδήποτε κύκλωμα, μπορούμε να ορίσουμε τον ελαττωμένο πίνακα πρόσπτωσης ως προς τον κόμβο αναφοράς. Συνήθως, ο κόμβος αναφοράς επιλέγεται να είναι ο κόμβος της γείωσης. Εφόσον το ρεύμα που ρέει στους κλάδους του κυκλώματος έχει συγκεκριμένη φορά, ο γράφος του κυκλώματος θα είναι κατευθυνόμενος και ο ελαττωμένος πίνακας πρόσπτωσης θα εξάγεται ανάλογα.

Μπορούμε με την χρήση του πίνακα πρόσπτωσης να εκφράσουμε τους νόμους του $Kirchhoff$, έτσι ώτε να προκύψουν διανυσματικές εξισώσεις. Ας θεωρήσουμε λοιπόν ένα κύκλωμα με $V = \{0, 1, 2, ..., n-1\}$ το σύνολο των n κόμβων (*nodes*) ή κορυφών (*vertices*), όπου 0 είναι ο κόμβος αναφοράς, και $E = \{e_1, e_2, ..., e_m\}$ το σύνολο των m κλάδων (*branches*) ή ακμών (*edges*) του κυκλώματος. Για το κύκλωμα αυτό ορίζουμε:

- Τον πίνακα πρόσπτωσης A του κυκλώματος ως προς την γείωση (τον κόμβο αναφοράς) με διάσταση:
$$dim\{A\} = (n-1) \times m.$$
- Το διάνυσμα στήλη των τάσεων των κλάδων του κυκλώματος $\vec{u}(t) = [u_1(t), u_2(t), ..., u_m(t)]^T$ με διάσταση: $dim\{\vec{u}(t)\} = m \times 1$.
- Το διάνυσμα στήλη των δυναμικών των κόμβων του κυκλώματος ως προς την γείωση $\vec{v}(t) = [v_1(t), v_2(t), ..., v_{n-1}(t)]^T$ με διάσταση $dim\{\vec{u}(t)\} = (n-1) \times 1$.
- Το διάνυσμα στήλη των ρευμάτων των κάδων του κυκλώματος $\vec{i}(t) = [i_1(t), i_2(t), ..., i_m(t)]^T$ με διάσταση $dim\{\vec{i}(t)\} = m \times 1$.

Ο νόμος τάσεων του $Kirchhoff$ (KVL) μπορεί να εκφραστεί με την εξίσωση:

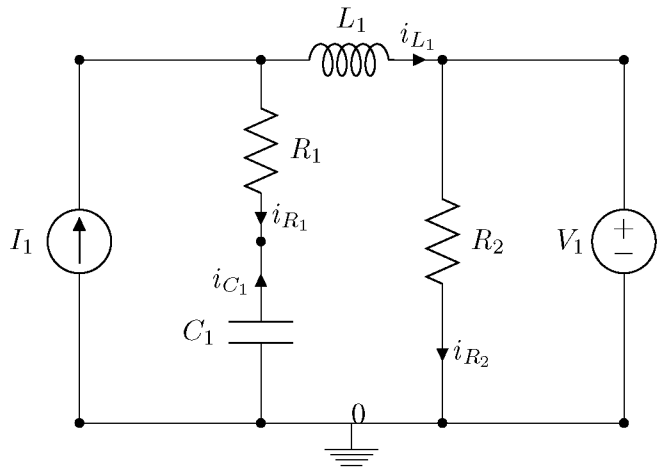
$$\vec{u}(t) = A^T \vec{v}(t).$$

Η σχέση αυτή εκφράζει οτι η τάση κατά μήκος κάθε κλάδου είναι ίση με την διαφορά δυναμικού των κόμβων-άκρων του κλάδου. Ο νόμος ρευμάτων του $Kirchhoff$ (KCL) εκφράζεται αντίστοιχα, μέσω της εξίσωσης:

$$A\vec{i}(t) = \vec{0}.$$

Η σχέση αυτή εκφράζει οτι το αλγεβρικό άθροισμα των ρευμάτων που προσπίπτουν σε κάθε κόμβο είναι ίσο με το μηδέν. Ας δούμε όμως τις εξισώσεις αυτές σε ένα παράδειγμα.

Ας θεωρήσουμε το παρακάτω κύκλωμα.



Ο πίνακας πρόσπτωσης του κυκλώματος ως προς την γείωση είναι:

$$A = \begin{bmatrix} -1 & +1 & +1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & +1 & +1 \end{bmatrix},$$

και επομένως, ο KVL θα είναι:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ +1 & 0 & -1 \\ +1 & -1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & +1 \\ 0 & 0 & +1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix},$$

και αντίστοιχα ο KCL θα είναι:

$$\begin{bmatrix} -1 & +1 & +1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & +1 & +1 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ i_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

2.5 Τροποποιημένη ανάλυση κόμβων

Η τροποποιημένη ανάλυση κόμβων είναι μια επέκταση της γνωστής μεθόδου κόμβων. Η μέθοδος των κόμβων χρησιμοποιείται για την εύρεση των τάσεων των κόμβων σε ένα ηλεκτρικό κύκλωμα. Αντίστοιχα, η τροποποιημένη ανάλυση κόμβων ($Modified\ Nodal\ Analysis\ -\ MNA$) υπολογίζει εκτός από τις τάσεις των κόμβων και το ρεύμα ορισμένων κλάδων του κυκλώματος. Τα m στοιχεία του κυκλώματος χωρίζονται σε δύο ομάδες χωρίζονται σε δύο ομάδες:

1. Τα στοιχεία της ομάδας 1 είναι στοιχεία των οποίων οι εξισώσεις μπορούν να γραφούν στην μορφή:

$$i_k(t) = g_k u_k(t) + c_k \frac{du_k(t)}{dt} + s_k(t).$$

Τα στοιχεία που εμπίπτουν στην ομάδα αυτή είναι οι αντιστάτες ($i = gu$), οι πυκνωτές ($i = cdu/dt$) και οι πηγές ρεύματος ($i = s$, όπου s γνωστή συνάρτηση).

2. Στοιχεία των οποίων οι εξισώσεις δεν μπορούν να γραφούν στην παραπάνω μορφή αποτελούν τα στοιχεία της ομάδας 2. Στην ομάδα αυτή εμπίπτουν οι πηγές τάσης και τα πηνία.

Υποθέτοντας λοιπόν ότι m_1 είναι τα στοιχεία της ομάδας 1 και m_2 είναι τα στοιχεία της ομάδας 2 (προφανώς $m = m_1 + m_2$) χωρίζουμε τον ελαττωμένο πίνακα πρόσπτωσης A και τα διανύσματα $\vec{u}(t)$ και $\vec{i}(t)$ σε δύο υποπίνακες και αντίστοιχα σε δύο υποδιανύσματα, τα οποία αντιστοιχούν στις δύο ομάδες των στοιχείων ως εξής:

$$A = \begin{bmatrix} A_1 & A_2 \end{bmatrix}, \quad \vec{u}(t) = \begin{bmatrix} \vec{u_1}(t) \\ \vec{u_2}(t) \end{bmatrix}, \quad \vec{i}(t) = \begin{bmatrix} \vec{i_1}(t) \\ \vec{i_2}(t) \end{bmatrix}$$

Οι δε διαστάσεις των υποπινάκων A_1 και A_2 θα είναι:

$$dim\{A_1\} = (n-1) \times m_1, \quad dim\{A_2\} = (n-1) \times m_2,$$

των υποδιανυσμάτων $\vec{u_1}(t)$ και $\vec{u_2}(t)$:

$$dim\{\vec{u_1}(t)\} = m_1 \times 1, \quad dim\{\vec{u_2}(t)\} = m_2 \times 1,$$

και των αντίστοιχα των υποδιανυσμάτων $\vec{i_1}(t)$ και $\vec{i_2}(t)$:

$$dim\{\vec{i_1}(t)\} = m_1 \times 1, \quad dim\{\vec{i_2}(t)\} = m_2 \times 1.$$

Με βάση τα παραπάνω ο KCL γράφεται ως εξής:

$$A\vec{i}(t) = \vec{0} \Leftrightarrow \begin{bmatrix} A_1 & A_2 \end{bmatrix} \begin{bmatrix} \vec{i_1}(t) \\ \vec{i_2}(t) \end{bmatrix} = 0 \Leftrightarrow A_1\vec{i_1}(t) + A_2\vec{i_2}(t) = \vec{0} \tag{2.1}$$

και αντίστοιχα ο KVL γράφεται:

$$\vec{u}(t) = A^T\vec{v}(t) \Leftrightarrow \begin{bmatrix} \vec{u_1}(t) \\ \vec{u_2}(t) \end{bmatrix} = \begin{bmatrix} A_1^T \\ A_2^T \end{bmatrix} \vec{v}(t) \Leftrightarrow \begin{cases} \vec{u_1}(t) = A_1^T\vec{v}(t) \\ \vec{u_2}(t) = A_2^T\vec{v}(t) \end{cases}$$

Άρα προκύπτει η εξίσωση για τα στοιχεία της ομάδας 1

$$\vec{u_1}(t) = A_1^T\vec{v}(t) \tag{2.2}$$

και αντίστοιχα η εξίσωση για τα στοιχεία της ομάδας 2

$$\vec{u_2}(t) = A_2^T\vec{v}(t). \tag{2.3}$$

Οι εξισώσεις των στοιχείων της ομάδας 1 γράφονται υπό μορφή πίνακα:

$$\vec{i_1}(t) = G\vec{u_1}(t) + C\frac{d\vec{u_1}(t)}{dt} + \vec{s_1}(t). \tag{2.4}$$

Ο πίνακας G με διάσταση $dim\{G\} = m_1 \times m_1$, είναι διαγώνιος με μη-μηδενικά στοιχεία στις θέσεις των αντιστατών και μηδενικά σις θέσεις των πυκνωτών και των πηγών ρεύματος. Ο πίνακας C με διάσταση $dim\{C\} = m_1 \times m_1$, είναι διαγώνιος με μη-μηδενικά στοιχεία στις θέσεις των

πυκνωτών και μηδενικά σις θέσεις των αντιστατών και των πηγών ρεύματος. Όπως προαναφέρθηκε, όταν ο k-οστός κλάδος φέρει αντίστατη η εξίσωση του θα είναι $i_k(t) = g_k u_k(t)$, ενώ όταν ο m-οστός κλάδος φέρει ένα πυκνώτη η εξίσωσή του θα είναι $i_m(t) = g_m du_m(t)/dt$. Τέλος, το διάνυσμα $\vec{s_1}$ με διάσταση $dim(\vec{s_1}) = m_1 \times 1$, έχει μη-μηδενικά στοιχεία στις θέσεις των πηγών ρεύματος και μηδενικά στις θέσεις των πυκνωτών και των αντιστατών.

Αντίστοιχα διαμορφώνονται και οι εξισώσεις των στοιχείων της ομάδας 2 σε μορφή πινάκων:

$$\vec{u_2}(t) = L\frac{d\vec{i_2}(t)}{dt} + \vec{s_2}(t). \tag{2.5}$$

Ο πίνακας L με διάσταση $dim\{L\} = m_2 \times m_2$, είναι διαγώνιος με μη-μηδενικά στοιχεία στις θέσεις των πηνίων και μηδενικά σις θέσεις των πηγών τάσης. Το διάνυσμα $\vec{s_2}$ με διάσταση $dim\{\vec{s_2}\} = m_2 \times 1$, έχει μη-μηδενικά στοιχεία στις θέσεις των πηγών τάσης και μηδενικά στις θέσεις των πηνίων.

Αντικαθιστώντας την εξίσωση 2.2 στην 2.4 και κατόπιν το αποτέλεσμα αυτής στην εξίσωση 2.1 προκύπτει:

$$A_1 G A_1^T \vec{v_1}(t) + A_1 C A_1^T \frac{d\vec{v_1}(t)}{dt} + A_2\vec{i_2}(t) = -A_1\vec{s_1}(t) \tag{2.6}$$

Επίσης, αντικαθιστώντας την εξίσωση 2.3 στην 2.5 προκύπτει:

$$A_2^T \vec{v}(t) - L\frac{d\vec{i_2}(t)}{dt} = \vec{s_2}(t) \tag{2.7}$$

Ο συνδυασμός των εξισώσεων 2.6 και 2.7 δίνει ένα σύστημα δίνει ένα σύστημα διαστάσεων:

$$[(n-1) + m_2] \times [(n-1) + m_2]$$

το οποίο το γράφουμε σε έναν επεκταμένο (σύνθετο) πίνακα ως εξής:

$$\begin{bmatrix} A_1 G A_1^T & A_2 \\ A_2^T & 0 \end{bmatrix} \begin{bmatrix} \vec{v_1}(t) \\ \vec{i_2}(t) \end{bmatrix} + \begin{bmatrix} A_1 C A_1^T & 0 \\ 0 & -L \end{bmatrix} \frac{d}{dt} \begin{bmatrix} \vec{v_1}(t) \\ \vec{i_2}(t) \end{bmatrix} = \begin{bmatrix} -A_1\vec{s_1}(t) \\ \vec{s_2}(t) \end{bmatrix}$$

2.5.1 Μεταβατική ανάλυση γραμμικών κυκλωμάτων (*transient analysis*)

Για την μεταβατική ανάλυση (*transient analysis*) ή ανάλυση χρονικής απόκρισης, το MNA σύστημα ενός γραμμικού κυκλώματος (με στοιχεία R, L, C και ανεξάρτητες πηγές) είναι αυτό που περιγράψαμε στην τελευταία εξίσωση. Θέτοντας τους πίνακες:

$$\tilde{G} = \begin{bmatrix} A_1 G A_1^T & A_2 \\ A_2^T & 0 \end{bmatrix} \quad , \quad \tilde{C} = \begin{bmatrix} A_1 C A_1^T & 0 \\ 0 & -L \end{bmatrix}$$

και τα διανύσματα:

$$\vec{x}(t) = \begin{bmatrix} \vec{v_1}(t) \\ \vec{i_2}(t) \end{bmatrix} \quad , \quad \vec{e}(t) \begin{bmatrix} -A_1\vec{s_1}(t) \\ \vec{s_2}(t) \end{bmatrix}$$

λαμβάνουμε ένα σύστημα γραμμικών διαφορικών εξισώσεων πρώτης τάξης με σταθερούς συντελεστές:

$$\tilde{G}\vec{x}(t) + \tilde{C}\frac{d\vec{x}(t)}{dt} = \vec{e}(t) \Leftrightarrow \tilde{G}\vec{x}(t) + \tilde{C}\dot{\vec{x}}(t) = \vec{e}(t) \tag{2.8}$$

Το διάνυσμα $\vec{e}(t)$ είναι το διάνυσμα των διεγέρσεων από τις πηγές τάσης και ρεύματος, ενώ το διάνυσμα $\vec{x}(t)$ είναι το διάνυσμα των αποκρίσεων (τάσεις κόμβων και ρεύματα κλάδων). Εάν καθοριστούν οι τιμές του διανύσματος $\vec{x}(t)$ σε μία χρονική στιγμή t_0 ως $\vec{x}(t_0) = \vec{x_0}$ (αρχική συνθήκη), τότε το πρόβλημα:

$$\begin{cases} \tilde{G}\vec{x}(t) + \tilde{C}\dot{\vec{x}}(t) = \vec{e}(t) \\ \vec{x}(t_0) = \vec{x_0} \end{cases}$$

ονομάζεται πρόβλημα αρχικών τιμών ($initial\ value\ problem\ -IVP-$) και υπό ορισμένες προϋποθέσεις έχει μοναδική λύση $\vec{x}(t)$ σε ένα διάστημα $[t_0,\ t_f]$ σε διακριτούς χρόνους:

$$t_0 < t_1 < t_2 < ... < t_m = t_f$$

και υπολογίζοντας μία προσέγγιση της λύσης $\vec{x}(t)$ σε κάθε διακριτή χρονική στιγμή t_k ($k = 1, 2, ..., m$) ξεκινώντας από την αρχική συνθήκη $\vec{x}(t_0) = \vec{x_0}$ (λύση DC σημείου λειτουργίας) και οδεύοντας διαδοχικάσε μεγαλύτερους χρόνους $t_1,\ t_2,\ ...,\ t_m$. Η τιμή $h_k = t_k - t_{k-1}$ καλείται χρονικό βήμα ή βήμα δειγματοληψίας την χρονική στιγμή t_k. Εάν τα χρονικά σημεία $t_0 < t_1 < t_2 < ... < t_m$ ισαπέχουν τότε $h_1 = h_2 = ... = h_m =\equiv h$ (οπότε και $t_k = t_0 + kh$) και το χρονικό βήμα είναι σταθερό.

Ο υπολογισμός της προσέγγισης $\vec{x}(t_k)$ για κάθε διακριτό χρόνο t_k ($k = 1, 2, ...m$) γίνεται μέσω μιας από τις ακόλουθες προσεγγίσεις της παραγώγου $\dot{\vec{x}}(t)$ στο σύστημα διαφορικών εξισώσεων.

1. Προσέγγιση $Backward\ Euler\ (BE)$ ή $Implicit\ Euler\ (IE)$:
   Σύμφωνα με αυτήν την προσέγγιση η παράγωγος προσεγγίζεται ως εξής:

   $$\frac{d\vec{x}(t_k)}{dt} \approx \frac{1}{h}[\vec{x}(t_k) - \vec{x}(t_{k-1})]$$

   και επομένως το γραμμικό σύστημα διαφορικών εξισώσεων της εξίσωσης 2.8 μετατρέπεται στο παρακάτω, αλγεβρικό γραμμικό σύστημα:

   $$(\tilde{G} + \frac{1}{h}\tilde{C})\vec{x}(t_k) = \vec{e}(t_k) + \frac{1}{h}\tilde{C}\vec{x}(t_{k-1})$$

   για $k = 1, 2, ...m$.

2. Προσέγγιση $Trapezoidal\ (TR)$ Σύμφωνα με αυτήν την προσέγγιση η παράγωγος προσεγγίζεται ως εξής:

   $$\frac{1}{2}[\frac{d\vec{x}(t_k)}{dt} + \frac{d\vec{x}(t_{k-1})}{dt}] \approx \frac{1}{h}[\vec{x}(t_k) - \vec{x}(t_{k-1})]$$

   και επομένως το γραμμικό σύστημα διαφορικών εξισώσεων της εξίσωσης 2.8 μετατρέπεται στο παρακάτω, αλγεβρικό γραμμικό σύστημα:

   $$\tilde{G}[\vec{x}(t_k) + \vec{x}(t_{k-1})] + \tilde{G}[\frac{\vec{x}(t_k)}{dt} + \frac{\vec{x}(t_{k-1})}{dt}] = \vec{e}(t_k) + \vec{e}(t_{k-1})$$

   ή ισοδύναμα:

   $$(\tilde{G} + \frac{2}{h}\tilde{C})\vec{x}(t_k) = \vec{e}(t_k) + \vec{e}(t_{k-1}) + (-\tilde{G} - \frac{2}{h}\tilde{C})\vec{x}(t_{k-1})$$

Η προσέγγιση $trapezoidal$ είναι πιο ακριβής και προσεγγίζει καλύτερα την πραγματική λύση $\vec{x}(t)$ για τις διακριτές στιγμές t_k. Ένα άλλο πλεονέκτημα έναντι της BE είναι οτι επιτρέπει μεγαλύτερα βήματα h_k σε μεταβλητή δειγματοληψία. Ωστόσο, σε απότομες μεταβολές των διεγέρσεων είναι λιγότερο ακριβής και καποιες φορές εμφανίζει το φαινόμενο "$ringing$".

2.5.2 Ανάλυση συνεχούς γραμμικών κυκλωμάτων ($DC\ analysis$)

Στην περίπτωση της DC ανάλυσης, όπου δεν υπάρχει χρονική μεταβολή, οι διεγέρσεις και οι αποκρίσεις θα είναι σταθερές, ως προς τον χρόνο συναρτήσεις, και επομένως η χρονική παράγωγός τους θα είναι μηδέν. Επομένως, το σύστημα της εξίσωσης 2.8 γίνεται:

$$\begin{bmatrix} A_1 G A_1^T & A_2 \\ A_2^T & 0 \end{bmatrix} \begin{bmatrix} \vec{v_1} \\ \vec{i_2} \end{bmatrix} = \begin{bmatrix} -A_1 \vec{s_1} \\ \vec{s_2} \end{bmatrix}. \tag{2.9}$$

Βλέπουμε λοιπόν πως είτε στην περίπτωση της $transient$ ανάλυσης, είτε στην περίπτωση της DC ανάλυσης καταλήγουμε στην επίλυση ενός γραμμικού συστήματος. Στο επόμενο κεφάλαιο θα μελετήσουμε αλγορίθμους για την επίλυση των γραμμικών συστημάτων.

Στην παρούσα εργασία θα επικεντρωθούμε στην DC ανάλυση και στην επίλυση του γραμμικού συστήματος της εξίσωσης 2.9, κυρίως όταν ο πίνακας του συστήματος είναι συμμετρικός και θετικά ορισμένος ($symmetric\ and\ positive\ defined\ -SPD-$). Ο πίνακας G είναι SPD όταν το κύκλωμα αποτελείται από στοιχεία της ομάδας 1, δηλαδή αντιστάτες, πυκνωτές και πηγές ρεύματος.

Σημειώνουμε στο σημείο αυτό οτι, ένας πίνακας A είναι συμμετρικός ($symmetric$) όταν ισχύει $A(i,j) = A(j,i)$ για $i \neq j$, δηλαδή όταν ο πίνακας ισούται με τον ανάστροφό του.

$$A = A^T \Leftrightarrow A\ is\ symmetric,\ \forall A\ \epsilon\ \Re^{nxn}.$$

Επίσης ένας πίνακας A είναι θετικά ορισμένος ($positive\ defined$) όταν η τετραγωνική μορφή είναι θετική για κάθε διάνυσμα εκτός του μηδενικού:

$$\vec{x}^T A \vec{x} > 0,\ \forall\ \vec{x}\epsilon\ \Re^n, \vec{x} \neq \vec{0} \Leftrightarrow A\ is\ positive\ defined.$$

Ένας άλλος ορισμός, για θετικά ορσμένο πίνακα, είναι οι ιδιοτιμές του να είναι θετικές.

Σε αυτήν την περίπτωση ο πίνακας $\tilde{G}$ θα είναι $\tilde{G} = A_1 G A_1^T$, το διάνυσμα των αγνώστων $\vec{x} = \vec{v}$ και το διάνυσμα δεξιού μέλους$\vec{b} = -A_1 \vec{s_1}$.

$$A_1 G A_1^T \vec{v} = -A_1 \vec{s_1} \Leftrightarrow \tilde{G}\vec{x} = \vec{b}$$

Ας θεωρήσουμε ένα κύκλωμα το οποίο αποτελείτε μόνο απο στοιχεία της ομάδας 1, και για το οποίο θέλουμε να κατασκευάσουμε το γραμμικό σύστημα για την DC προσομοίωση του. Στο DC οι πυκνωτές συμπεριφέρονται ως ανοιχτοκυκλώματα και επομένως δεν μας απασχολούν. Ο πίνακας G κατασκευάζεται λοπίν από τις συνεισφορές των ηλεκτρικών αγωγιμοτήτων των αντιστατών. Έτσι λοιπόν:

1. Τα διαγώνια στοιχεία ($diagonal\ elements$) $\tilde{G}(k,k)$ θα είναι το άθροισμα των ηλεκτρικών αγωγιμοτήτων των αντιστατών που συνδέονται στον κόμβο k

2. Τα εκτός διαγωνίου στοιχεία ($off-diagonal\ elements$) $\tilde{G}(k,i)$ και $\tilde{G}(i,k)$ θα είναι το αρνητικό άθροισμα των ηλεκτρικών αγωγιμοτήτων $g_{i,k}$ των αντιστατών που συνδέονται στους κόμβους i και k.

2.6 Αραιοί πίνακες

Από την ανάλυση της προηγούμενης παραγράφου για την κατασκευή του πίνακα $\tilde{G}$ παρατηρούμε πως ο πίνακας θα έχει πολλά μηδενικά στοιχεία. Για την αποδοτική αποθηκευσή τους λοιπόν, χρειαζόμαστε κατάλληλες δομές δεδομένων.

Ένας πίνακας $A \epsilon \Re^{nxn}$ με n_z αριθμό μη-μηδενικών στοιχείων θα ονομάζεται αραιός όταν:

$$n_z = O(n)$$

Υπολογιστικά, ο αραιός πίνακας δηλώνει μια δομή δεδομένων που αναπαριστά έναν πίνακα, που αποθηκεύονται και συμμετέχουν στους υπολογισμούς μόνο τα μη-μηδενικά στοιχεία. Οι πιό γνωστές δομές δεδομένων για την αποθήκευση των αραιών πινάκων πινάκων είναι οι ακόλουθες:

1. Μορφή *triplet*:
   Στην μορφή *triplet* χρημοποιούναι τρία διανύσματα (*arrays*) μεγέθους n_z. Τα δυό διανύσματα είναι τύπου *int* και απόθηκεύουν τις συντεταγμένες των εκάστοτε μη-μηδενικών στοιχείων, ενώ στο τρίτο διάνυσμα αποθηκεύονται οι τιμές αυτών. Τα μη-μηδενικά στοιχεία μπορούν να βρίσκονται εντός των διανυσμάτων με αυθαίρετη σειρά.

   Για παράδειγμα, ας θεωρήσουμε τον πίνακα:

$$A = \begin{bmatrix} 4.5 & 0 & 3.2 & 0 \\ 3.1 & 2.9 & 0 & 0.9 \\ 0 & 1.7 & 3.0 & 0 \\ 3.5 & 0.4 & 0 & 1.0 \end{bmatrix}$$

   Για την αναπαράσταση σε μορφή *triplet* θεωρούμε το διάνυσμα *int* $i[\,]$ το οποιο θα περιέχει τους δείκτες γραμμών των μη-μηδενικών στοιχείων (*row indices*), το διάνυσμα *int* $j[\,]$ το οποιο θα περιέχει τους δείκτες στηλών των στοιχείων αυτών (*column indices*) και το διάνυσμα *double* $x[\,]$ το οποίο θα περιέχει τις τιμές των αντίστοιχων στοιχείων. Επομένως, θα είναι:

$$int\ i[\,] = \{2, 1, 3, 0, 1, 3, 3, 1, 0, 2\}$$

$$int\ j[\,] = \{2, 0, 3, 2, 1, 0, 1, 3, 0, 1\}$$

$$double\ x[\,] = \{3.0, 3.1, 1.0, 3.2, 2.9, 3.5, 0.4, 0.9, 4.5, 1.7\}$$

   Σημειώνουμε στο σημείο αυτό ότι η αρίθμηση των θέσεων του πίνακα ξεκινά από το 0 ($zero-offset$), όπως και στις γλώσσες προγραμματισμού C και $C++$ τις οποίες χρησιμοποιούμε για τον προγραμματισμό των αλγορίθμων.

2. Μορφή *compressed row*:
   Στην μορφή *compressed row* για την αναπαράσταση ενός πίνακα $A \epsilon \Re^{nxn}$ χρησιμποιούνται τρία δανύσματα (*arrays*). Ένα διάνυσμα *int* $j[\,]$ μεγέθους n_z το οποίο περιέχει τους δείκτες στηλών (*column indices*) των μη-μηδενικών στοιχείων. Ένα διάνυσμα *int* $p[\,]$ μεγέθους $n+1$ το οποίο περιέχει τους δείκτες σε γραμμές (*row pointers*) των μη-μηδενικών στοιχείων. Το τρίτο διάνυσμα, *double* $x[\,]$ μεγέθους n_z με τις τιμές των μη-μηδενικών στοιχείων. Το διάνυσμα p είναι τέτοιο ώστε για την γραμμή i τα *column indices* αποθηκεύονται στα $j[p[i]],\ j[p[i]+$

1], ..., $j[p[i+1]-1]$ και τα αντίστοιχα μη-μηδενικά στοιχεία στα $x[p[i]]$, $x[p[i]+1]$, ..., $x[p[i+1]-1]$. Έτσι λοιπόν για τον πίνακα του παραδείγματός μας η αποθήκευσή του σε μορφή *compressed row* θα είναι:

$$int\ p[\ ] = \{0, 2, 5, 7, 10\}$$

$$int\ j[\ ] = \{0, 2, 0, 1, 3, 1, 2, 0, 1, 3\}$$

$$double\ x[\ ] = \{4.5, 3.2, 3.1, 2.9, 0.9, 1.7, 3.0, 3.5, 0.4, 1.0\}$$

Να σημειώσουμε οτι και εδώ διατηρείτε το $zero - offset$ και οτι η 10-η γραμμή δεν υπάρχει, αλλά μας βοηθάει στους υπολογισμούς ($j < 10$).

3. Μορφή *compressed column*:
   Στην μορφή *compressed column* για την αναπαράσταση ενός πίνακα $A \in \Re^{nxn}$ χρησιμποιούνται τρία δανύσματα (*arrays*). Ένα διάνυσμα $int\ i[\ ]$ μεγέθους n_z το οποίο περιέχει τους δείκτες γραμμών (*row indices*) των μη-μηδενικών στοιχείων. Ένα διάνυσμα $int\ p[\ ]$ μεγέθους $n+1$ το οποίο περιέχει τους δείκτες σε στείλες (*column pointers*) των μη-μηδενικών στοιχείων. Το τρίτο διάνυσμα, $double\ x[\ ]$ μεγέθους n_z με τις τιμές των μη-μηδενικών στοιχείων.

   Το διάνυσμα p είναι τέτοιο ώστε για την στήλη j τα *row indices* αποθηκεύονται στα $i[p[j]]$, $i[p[j]+1]$, ..., $i[p[j+1]-1]$ και τα αντίστοιχα μη-μηδενικά στοιχεία στα $x[p[j]]$, $x[p[j]+1]$, ..., $x[p[j+1]-1]$. Έτσι λοιπόν για τον πίνακα του παραδείγματός μας η αποθήκευσή του σε μορφή *compressed column* θα είναι:

$$int\ p[\ ] = \{0, 3, 6, 8, 10\}$$

$$int\ i[\ ] = \{0, 1, 3, 1, 2, 3, 0, 2, 1, 3\}$$

$$double\ x[\ ] = \{4.5, 3.1, 3.5, 2.9, 1.7, 0.4, 3.2, 3.0, 0.9, 1.0\}$$

   Να σημειώσουμε οτι και εδώ διατηρείτε το $zero - offset$ και οτι η 10-η στήλη δεν υπάρχει, αλλά μας βοηθάει στους υπολογισμούς ($j < 10$).

Στην εργασία αυτή χρησιμοποιούμε την βιβλιοθήκη *Csparse* στην οποία υλοποιούνται οι δομές *triplet* και *compressed column* καθώς και διάφοροι αλγόριθμοι επί των δομών αυτών. Αρχικά, χρησιμοποιούμε την δομή *triplet* η οποία είναι κατάλληλη για την εισαγωγή των μη-μηδενικών στοιχείων του αραιού πίνακα και στην συνέχεια, μέσω κατάλληλων συναρτήσεων, ο πίνακας αποθηκεύεται σε μορφή *compressed column* η οποία είναι κατάλληλη για πράξεις πινάκων και διανυσμάτων.

2.7 Αρχεία περιγραφής κυκλωμάτων

Το αρχείο εισόδου του προσομοιωτή είναι ένα αρχείο περιγραφής κυκλώματος (*netlist*). Το αρχείο περιγραφής του κυκλώματος περιέχει όλες τις πληροφορίες που χρειάζομαστε για τα κυκλωματικά στοιχεία και τον τρόπο με τον οποίο αυτά συνδέονται. Κάθε γραμμή του αρχείου εισόδου περιγράφει ένα κυκλωματικό στοιχείο για το οποίο αναγράφονται ο τύπος του, το όνομα του, ο κόμβος στον οποίο προσπίπτει ο θετικός, ο κόμβος στον οποίο προσπίπτει ο αρνητικός ακροδέκτης του, και η τιμή του όπως μετράται στο διεθνές σύστημα μονάδων (*S.I.*).

Έτσι λοιπόν τα κυκλωματικά στοιχεία που υποστηρίζονται θα ακολουθούν την παρακάτω μορφή εντός ενός αρχείου περιγραφής κυκλώματος:

- Ανεξάρτητη πηγή τάσης: $V < name > < + > < - > < value >$
- Ανεξάρτητη πηγή ρεύματος: $I < name > < + > < - > < value >$
- Αντιστάτης: $R < name > < + > < - > < value >$
- Πυκνωτής: $C < name > < + > < - > < value >$
- Πηνίο: $L < name > < + > < - > < value >$

Εκτός από τα κύκλωματικά στοιχεία το αρχείο περιγραφής κυκλώματος προσδιορίζει την μέθοδο με την οποία θα γίνει η επίλυση καθώς και επιπλέον πληροφορίες για το κύκλωμα. Αυτές οι πληροφορίες δίνονται στο τμήμα έπιλογών', δηλαδή σε μία γραμμή η οποία ξεκινά με *.options* ή *.op*. Στο πρόγραμμα μας θα χρησιμοποιήσουμε το τμήμα επιλογών για να προσδιορίσουμε τον εκάστοτε προρυθμιστή κατάστασης που επιθυμούμε να χρησιμοποιήσουμε.

Για παράδειγμα στο τμήμα επιλογών μπορεί να υπάρχει η δήλωση:

$$.options\ sparse\ iter\ SPD\ precondAST$$

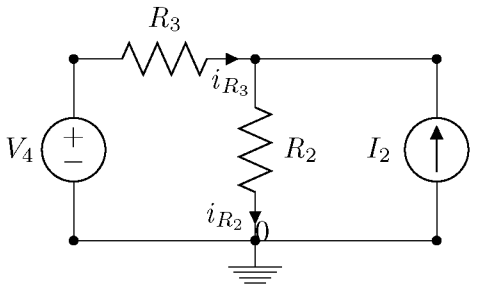
Η επιλογή *sparse* δηλώνει να χρησιμοποιηθεί η δομή αραιών πινάκων (ενδεχομένως γιατί το κύκλωμα είναι αρκετά μεγάλο). Η επιλογή *iter* δηλώνει την χρήση επαναληπτικής μεθόδου για την επίλυση του γραμμικού συστήματος και η επιλογή SPD μας λέει πως ο πίνακας που θα προκύψει θα είναι συμμετρικός και θετικά ορισμένος. Υπό αυτές τις συνθήκες θα χρησιμοποιήσουμε ως επιλυτή την μέθοδο συζυγών κλίσεων *conjugate gradient*. Τέλος, η δήλωση *precondAST* δηλώνει οτι ως προρυθμιστής κατάστασης θα είναι ο πίνακας που θα προκύψει από ένα επαυξημένο επικαλύπτον δέντρο χαμηλής έκτασης του γράφου που ορίζουν οι αντιστάτες του κυκλώματος.

Ας συνοψίσουμε με ένα απλό παράδειγμα όμως όσα είδαμε έως τώρα. Αρχικά ας θεωρήσουμε τα ακόλουθο αρχείο περιγραφής κυκλώματος:

$R1$	2	0	2
$I2$	0	2	2
$R3$	1	2	4
$V4$	1	0	3

Το κύκλωμα που περιγράφει αυτό το *netlist* είναι το ακόλουθο:

Ας θεωρήσουμε το παρακάτω κύκλωμα.



Το σύστημα που θα διαμορφωθεί για την DC προσομοίωση του κυκλώματος σύμφωνα με την τροποποιημένη ανάλυση κόμβων θα είναι:

$$\begin{bmatrix} \frac{1}{4} & -\frac{1}{4} & 1 \\ -\frac{1}{4} & \frac{1}{2}+\frac{1}{4} & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \\ \\ \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 3 \end{bmatrix}.$$

Σε μορφή *compressed column* και διατηρώντας το $zero-offset$ τα διανύσματα p, i και x θα διαμορφωθούν ως εξής:

$$p[\,] = \{0, 3, 5, 6\},$$

$$i[\,] = \{0, 1, 2, 0, 1, 0\},$$

$$x[\,] = \{\frac{1}{4}, -\frac{1}{4}, 1, -\frac{1}{4}, \frac{3}{4}, 1\}.$$

Κεφάλαιο 3

Αλγόριθμοι επίλυσης γραμμικών συστημάτων

Όπως είδαμε στο προηγούμενο κεφάλαιο είτε εκτελούμε DC προσομοίωση είτε $transient$ προσομοίωση ενός κυκλώματος, καταλήγουμε σε ένα γραμμικό σύστημα.

$$A\vec{x} = \vec{b}$$

Στο κεφάλαιο αυτό θα μελετήσουμε αλγορίθμους επίλυσης γραμμικών συστημάτων. Οι αλγόριθμοι επίλυσης γραμμικών συστημάτων χωρίζονται σε δύο κατηγορίες: στις άμεσες μεθόδους, οι οποίες τερματίζουν σε προκαθορισμένο αριθμό πράξεων και στις επαναληπτικές μεθόδους, οι οποίες δημιουργούν μία ακολουθία προσεγγίσεων της λύσης μέχρι κάποια σύγκλιση:

$$\vec{x}^{(0)},\ \vec{x}^{(1)},\ \vec{x}^{(2)},\ ...,\ \vec{x}^{(n)}$$

Σημειώνουμε οτι για να επιλύεται το γραμμικό σύστημα $A\vec{x} = \vec{b}$ πρέπει ο πίνακας A να είναι αντιστρέψιμος. Για να είναι αντιστρέψιμος ο πίνακας και συνεπώς επιλύσιμο το κύκλωμα, πρέπει να μην υπάρχει:

1) βρόγχος από πηγές τάσεις που το άθροισμά τους δεν είναι μηδέν. Σε αυτή την περίπτωση δεν ισχύει ο KVL και επομένως το κύκλωμα δεν είναι επιλύσιμο.

2) ομάδα διαχωρισμού (*cut set*). Η ομάδα διαχωρισμού αποτελείτε από πηγές ρεύματος που αν αφαιρεθούν χωρίζουν το κύκλωμασε δύο ξεχωριστά -μη συνδεδεμάνα- μέρη.

3.1 Άμεσες (*direct*) μέθοδοι επίλυσης γραμμικών συστημάτων

Οι άμεσες μέθοδοι είναι η επίλυση μέσω παραγοντοποίσης LU και επίλυση μέσω παραγοντοποίηση $Cholesky$.

3.1.1 Παραγοντοποίηση LU

Εάν $A \epsilon \Re^{nxn}$ τότε υπάρχουν μοναδικοί πίνακες L (μοναδικός κάτω τριγωνικός) και U (μοναδικός πάνω τριγωνικός):

$$L = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ l_{31} & l_{32} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \dots & 1 \end{bmatrix}, \quad U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & u_{nn} \end{bmatrix}$$

τέτοιοι ώστε $PA = LU$ όπου $P \epsilon \Re^{nxn}$ είναι πίνακας μετάθεσης (*permutation matrix*). Ο πίνακας P προκύπτει από τον μοναδιαίο $n \times n$ πίνακα I_n με εναλλαγές γραμμών. Το γινόμενο PA προκύπτει από τον A με τις ίδιες εναλλαγές γραμμών. Για την παραγαντοποίηση LU χρησιμοποιείται ο αλγόριθμος απαλοιφής *Gauss* (*Gaussian elimination*) ο οποιος μας δίνει τους πίνακες L και U. Η μετάθεση γραμμών χρησιμοποιείται για να αποφευχθεί η διαίρεση με το μηδέν (ή η απώλεια αριθμητικής ακρίβειας). Έτσι, κατά το τρέχον βήμα γίνεται εναλλαγή μη την γραμμή (εξίσωση) με το μαγαλύτερο (κατά απόλυτη τιμή) διαγώνιο στοιχείο.

Παρακάτω ακολουθεί ένα παράδειγμα εκτέλεσης του αλγορίθμου απαλοιφής *Gauss* (παραγοντοποίησης LU):

Το αρχικό σύστημα $A\vec{x} = \vec{b}$ είναι ισοδύναμο με το σύστημα γραμμικών εξισώσεων:

$$a_{11}x_1 + a_{12}x_2 + ... + a_{1n} = b_1$$

$$a_{21}x_1 + a_{22}x_2 + ... + a_{2n} = b_2$$

$$... \ ... \ ...$$

$$a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn} = b_n$$

Ας υποθέσουμε οτι ο οδηγός (*pivot*) του πρώτου βήματος είναι το στοιχείο a_{11}. Τότε μετά το πρώτο βήμα της απαλοιφής οι εξισώσεις γίνονται:

$$a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1$$

$$0x_1 + [a_{22} - a_{12}(\frac{a_{21}}{a_{11}})]x_2 + ... + [a_{2n} - a_{1n}(\frac{a_{21}}{a_{11}})]x_n = b_2 - b_1(\frac{a_{21}}{a_{11}})$$

$$... \ ... \ ...$$

$$0x_1 + [a_{n2} - (\frac{a_{n1}}{a_{11}})]x_2 + ... + [a_{nn} - (\frac{a_{n1}}{a_{11}})]x_n = b_n - b_1(\frac{a_{n1}}{a_{11}})$$

Ισοδύναμα, σε γραφή πινάκων, το αρχικό σύστημα είναι:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & \dots & a_{3n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ \vdots \\ b_n^{(1)} \end{bmatrix}$$

Μετά το πρώτο βήμα της απαλοιφής $Gauss$ το σύστημα γίνεται:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & \dots & a_{3n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix}$$

Έτσι λοιπόν, μετά από $(k-1)$ βήματα το σύστημα θα είναι:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1k}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \dots & a_{2k}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ 0 & 0 & \dots & a_{nk}^{(k)} & \ddots & a_{nn}^{(k)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_k^{(k)} \\ \vdots \\ b_n^{(k)} \end{bmatrix}$$

Και τελικά μετά από $(n-1)$ βημάτα προκύπτει το τελικό σύστημα του πίνακα U:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_1^{(2)} \\ \vdots \\ b_1^{(n)} \end{bmatrix}$$

Η ακολουθία αλλαγών στο διάνυσμα δεξιού μέλους $\vec{b}$ είναι:

step	*element* b_i
1	$b_i^{(1)} = b_i$ (αρχική τιμή)
2	$b_i^{(2)} = b_i^1 - (\frac{a_{i1}^{(1)}}{a_{11}^{(1)}})b_1^{(1)}$
$\vdots$	$\vdots$
κ	$b_i^{(k+1)} = b_i^k - (\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}})b_k^{(k)}$
$\vdots$	$\vdots$
ν	$b_i^{(n)} = b_i^{n-1} - (\frac{a_{n,n-1}^{(n-1)}}{a_{n-1,n-1}^{(n-1)}})b_{n-1}^{(n-1)}$

Το στοιχείο b_i δεν αλλάζει μετά το βήμα $(i-1)$. Αναπτύσσοντας λοιπόν την αναδρομή για την τελική μορφή του $b_i^{(i)}$ του b_i μετά από i βήματα, προκύπτει:

$$b_i^{(i)} = b_i - \sum_{j=1}^{i-1}(\frac{a_{ij}^j}{a_{jj}^{(j)}})b_j^{(j)} \Leftrightarrow b_i^{(i)} + \sum_{j=1}^{i-1}(\frac{a_{ij}^j}{a_{jj}^{(j)}})b_j^{(j)} = b_i^{(i)}$$

το οποίο γράφεται υπό μορφή πίνακα:

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ (\frac{a_{21}^{(1)}}{a_{11}^{(1)}}) & 1 & 0 & \dots & 0 \\ (\frac{a_{31}^{(1)}}{a_{11}^{(2)}}) & (\frac{a_{32}^{(1)}}{a_{22}^{(2)}}) & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (\frac{a_{n1}^{(1)}}{a_{11}^{(1)}}) & (\frac{a_{n2}^{(2)}}{a_{22}^{(2)}}) & (\frac{a_{n3}^{(3)}}{a_{33}^{(3)}}) & \dots & 1 \end{bmatrix} \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(3)} \\ \vdots \\ b_n^{(n)} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}.$$

Πολλαπλασιάζοντας από αριστερά την εξίσωση () με L προκύπτει:

$$LU\vec{x} = L \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(n)} \end{bmatrix} = \vec{b}$$

Άρα τελικά έχουμε το κάτω τριγωνικό σύστημα του πίνακα U:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \dots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & a_{23}^{(2)} & \dots & a_{2n}^{(2)} \\ & & a_{33}^{(3)} & \dots & a_{3n}^{(3)} \\ & & & \ddots & \vdots \\ & & & & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(3)} \\ \vdots \\ b_n^{(n)} \end{bmatrix}$$

όπου η το διάνυσμα δεξιού μέλους είναι η λύση του άνω τριγωνικού συστήματος του πίνακα L:

$$\begin{bmatrix} 1 & & & \\ (\frac{a_{21}^{(1)}}{a_{11}^{(1)}}) & 1 & & \\ \vdots & \vdots & \ddots & \\ (\frac{a_{n1}^{(1)}}{a_{11}^{(1)}}) & (\frac{a_{n2}^{(2)}}{a_{22}^{(2)}}) & \dots & 1 \end{bmatrix} \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(3)} \\ \vdots \\ b_n^{(n)} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

Παρακάτω παραθέτουμε τον αλγόριθμο παραγοντοποίησης LU για έναν $n \times n$ πίνακα A:

Algorithm 1 LUFactorization

Data: A
Result: L,U
for *k = 1, 2, ..., n* **do**
  x = |a_{kk}|
  for *i = k, ..., n* **do**
    if |*aik*| > *x* **then**
      m = i
    end
  end
  INTERCHANGE_ROWS(k, m)
  for *i = k+1, ..., n* **do**
    $a_{ik} = l_{ik} = \frac{a_{ik}}{a_{kk}}$
  end
  for *i = k+1, ..., n* **do**
    for *j = k+1, ..., n* **do**
      $a_{ij} = a_{ij} - l_{ik}a_{kj}$
    end
  end
end

Σημειώνουμε οτι κατά την μερκή οδήγηση πρέπει να υπάρχει αναλλαγή και των σοιχείων b_k και b_m του δεξιού διανύσματος $\vec{b}$, ταυτόχρονα με την εναλλαγή γραμμών k και m του A.

$$A\vec{x} = \vec{b} \Leftrightarrow PA\vec{x} = P\vec{b} \Leftrightarrow LU\vec{x} = P\vec{b}$$

Η πολυπλοκότητα της παραγοντοποίησης LU είναι $O(n^3)$.

Εάν $Q\ \epsilon\Re^{nxn}$ είναι πίνακας μετάθεσης που προκύπτει από τον $n \times\ n$ ταυτοτικό πίνακα I_n με εναλλαγές στηλών, τότε ο AQ προκύπτει από τον A με τις ίδιες εναλλαγές στηλών. Ο πίνακας PAQ εμπεριέχει εναλλαγές γραμμών και στηλών. Η διαδικασία αυτή ονομάζεται ολική οδήγηση ($full\ pivoting$).

Στους πυκνούς πίνακες δεν εφαρμόζεται ολική οδήγηση, καθώς δεν βελτιώνει την αριθμητική ακρίβεια. Σε αραιούς πίνακες πραγματοποιούνται εναλλαγές γραμμών (πίνακας P) για διατήρηση αριθμητικής ακρίβειας και αποφυγή διαίρεσης με το μηδέν, καθώς και εναλλαγές στηλών (πίνακας Q) ώστε να προκύψουν οι L και U όσο πιο αραιοί γίνεται (λόγω εισαγωγής νέων μη-μηδενικών στοιχείων -ή $fill - ins$-).

3.1.2 Παραγοντοποίηση $Cholesky$

Εάν $A\ \epsilon\ \Re^{nxn}$ είναι πίνακας συμμετρικός ($A = A^T$) και θετικά ορισμένος ($\vec{x}^T A\vec{x} > 0, \forall\vec{x}\ \epsilon\ \Re^n, \vec{x} \neq \vec{0}$) ($SPD$) τότε υπάρχει μοναδικός κάτω τριγωνικός πίνακας L τέτοιος ώστε $A = LL^T$ (χωρίς πίνακα μετάθεσης). Ο αλγόριθμος της παραγοντοποίησης $Cholesky$ είναι:

Algorithm 2 CholeskyFactorization

Data: A
Result: L
for *k = 1, 2, ..., n* **do**
  $l_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2}$
  for *i=k+1,...,n* **do**
    $a_{ik} = l_{ik} = \frac{1}{l_{kk}}(a_{ki} - \sum_{j=1}^{k-1} l_{ij} l_{kj})$
  end
end

Η παραγοντοποίηση $Cholesky$ δεν απαιτεί μεταθέσεις γραμμών (μερική οδήγηση). Ο αλγόριθμος αποτυγχάνει εάν ο πίνακας A δεν είναι SPD γιατί για κάποιο k θα είναι: $a_{kk} < \sum_{j=1}^{k-1} l_{kj}^2$.

3.1.3 Αλγόριθμοι επίλυσης τριγωνικών συστημάτων

Αφού λοιπόν ο πίνακας A παραγοντοποιηθεί -είτε με παραγοντοποίηση $Cholesky$, αν είναι SPD, είτε με παραγοντοποίηση LU σε διαφορετική περίπτωση- η επίλυση του γραμμικού συστήματος έγκειται στην επίλυση ενός κάτω τριγωνικού συστήματος και ενός άνω τρογωνικού συστήματος. Για παράδειγμα το σύστημα $LU\vec{x} = P\vec{b}$ ανάγεται στην επίλυση των τριγωνικών συσημάτων $L\vec{y} = \vec{b}$ και $U\vec{x} = \vec{y}$. Ο αλγόριθμος για την επίλυση του κάτω τριγωνικού συστήματος $L\vec{y} = \vec{b}$ είναι ο αλγόριθμος εμπρόσθιας αντικατάστασης ($forward\ substitution$), ο οποίος δίδεται παρακάτω:

Algorithm 3 ForwardSubstitution

Data: L, $\vec{b}$
Result: $\vec{y}$
for *k = 1, 2, ..., n* **do**
  for *j = 1, ..., k-1* **do**
    $b_k = b_k - l_{kj} y_j$
  end
  $y_k = \frac{b_k}{l_{kk}}$
end

Αντίστοιχα ο αλγόριθμος για την επίλυση του άνω τριγωνικού συστήματος $U\vec{x} = \vec{y}$ είναι ο αλγόριθμος οπίσθιας αντικατάστασης ($backward\ substitution$), ο οποίος δίδεται παρακάτω:

Algorithm 4 BackwardSubstitution

Data: U, $\vec{y}$
Result: $\vec{x}$
for *k=n,...,1* **do**
  for *j = k+1,...,n* **do**
    $y_k = y_k - u_{kj} x_j$
  end
  $x_k = \frac{y_k}{u_{kk}}$
end

3.2 Επαναληπτικές (*iterative*) μέθοδοι επίλυσης γραμμικών συστημάτων

Όπως προαναφέραμε, οι επαναληπτικές μέθοδοι επίλυσης δημιουργούν μία ακολουθία διαδοχικών προσεγγίσεων της λύσης $\vec{x}$ μέχρι κάποια σύγκλιση:

$$\vec{x^{(0)}}, \vec{x^{(1)}}, \vec{x^{(2)}}, ..., \vec{x^{(n)}}$$

Θα αναλύσουμε τον αλγόριθμο της μεθόδου των συζυγών κλίσεων (*conjugate gradient* $-cg-$) για επίλυση συστημάτων με συμμετρικούς και θετικά ορισμένους πίνακες και την επέκταση αυτού, τον αλγόριθμο $bi-cg$, για την επίλυση γενικών συστημάτων.

3.2.1 Μέθοδος συζυγών κλίσεων (*conjugate gradient*)

Έστω $A \epsilon \Re^{nxn}$ πίνακας συμμετρικός και θετικά ορισμένος (SPD), δηλαδή $A = A^T$ και $\vec{x}^T A\vec{x} > 0, \forall\ \vec{x}\ \epsilon\Re^n, \vec{x} \neq \vec{0}$, και έστω η συνάρτηση n-μεταβλητών:

$$f(\vec{x}) = \frac{1}{2}\vec{x}^T A\vec{x} - \vec{b}^T\vec{x} \quad ,\ \vec{x}\ \epsilon\ \Re^n$$

Η $f(\vec{x})$ παρουσιάζει κρίσιμο σημείο (*critical point*) στο $\vec{x}^*$ όπου:

$$\vec{\nabla} f(\vec{x})|_{\vec{x}=\vec{x}^*} = \vec{0} \ \Leftrightarrow$$

$$(\frac{\partial f(\vec{x})}{\partial x_1}|_{\vec{x}=\vec{x^*}}, \frac{\partial f(\vec{x})}{\partial x_2}|_{\vec{x}=\vec{x^*}}, ..., \frac{\partial f(\vec{x})}{\partial x_n}|_{\vec{x}=\vec{x^*}}) = \vec{0}$$

Η k-οστή συνιστώσα του διανύσματος $\vec{\nabla} f(\vec{x})$ θα είναι:

$$(\vec{\nabla} f(\vec{x}))|_k = \frac{\partial f(\vec{x})}{\partial x_k} = \frac{\partial}{\partial x_k}[\frac{1}{2}\vec{x}^T A\vec{x} - \vec{b}^T\vec{x}] =$$

$$= \frac{1}{2}\frac{\partial}{\partial x_k}[\sum_{i=1}^{n}(x_i\sum_{j=1}^{n}a_{ij}x_j)] - \frac{\partial}{\partial x_k}[\sum_{i=1}^{n}b_i x_i] =$$

$$= \frac{1}{2}[\sum_{j=1}^{n}a_{kj}x_j + \sum_{i=1}^{n}x_i a_{ik}] - b_k =$$

και επειδή ο A είναι συμμετρικός, θα ισχύει $a_{ij} = a_{ji}$ και άρα:

$$= \sum_{j=1}^{n}a_{kj}x_j - b_k = (A\vec{x} - b)_k$$

Επομένως λοιπόν είναι $\vec{\nabla} f(\vec{x}) = A\vec{x} = \vec{b}$ και η $f(\vec{x})$ έχει κρίσιμο σημείο στο $\vec{x}^*$ με:

$$\vec{\nabla} f(\vec{x})|_{\vec{x}=\vec{x}^*} = \vec{0} \Leftrightarrow A\vec{x} - \vec{b} = \vec{0} \Leftrightarrow A\vec{x} = \vec{b}$$

Το μοναδικό κρίσιμο σημείο είναι σημείο ελαχίστου, καθώς ο Εσσιανός πίνακας ($Hessian\ matrix$) H της $f(\vec{x})$ έχει στοιχεία:

$$h_{ij} = \frac{\partial^2}{\partial x_i \partial x_j}[f(\vec{x})] = \frac{\partial}{\partial x_j}[\frac{\partial}{\partial x_i}(f(\vec{x}))] = \frac{\partial}{\partial x_j}[(A\vec{x} - \vec{b})_i] = a_{ij}$$

Δηλαδή $H = A$, όπου ο A είναι SPD. Επομένως, και ο H είναι SPD πίνακας. Εάν βρούμε έναν αλγόριθμο για την εύρεση του ελαχίστου της συνάρτησης $f(\vec{x}) = \frac{1}{2}\vec{x}^T A\vec{x} - \vec{b}^T\vec{x}$ θα βρούμε και την λύση του συστήματος $A\vec{x} = \vec{b}$.

Η $conjugate\ gradient$ είναι μια απαναληπτική μέθοδος ελαχιστοποίησης της συνάρτησης: $f(\vec{x}) = \frac{1}{2}\vec{x}^T A\vec{x} - \vec{b}^T\vec{x}$. Σε κάθε βήμα υπολογίζεται μια κατεύθυνση αναζήτησης ($search\ direction$) $\vec{\rho}^{(i)}$ στο χώρο $\Re^n$ η οποία είναι ορθογώνια ($A-orthogonal$) ή $conjugate$ στην κατεύθυνση αναζήτησης του προηγούμενου βήματος $\vec{\rho}^{(i-1)}$.Εξ' ορισμού:

$$\vec{\rho}^{(i)^T} A\vec{\rho}^{(i-1)} = 0 \Leftrightarrow \vec{\rho}^{(i)}, \vec{\rho}^{(i-1)}\ A - orthogonal/conjugate$$

Έτσι, η τρέχουσα προσέγγιση $\vec{x}^{(i-1)}$ μετακινείται προς την $\vec{\rho}^{(i)}$ και γίνεται:

$$\vec{x}^{(i)} = \vec{x}^{(i-1)} + a_i\vec{\rho}^{(i)}$$

Το βήμα μετακίνησης a_i ελαχιστοποιεί την συνάρτηση $g(a_i) = f(\vec{x}^{(i-1)} + a_i\vec{\rho}^{(i)})$ επί όλων των a_i.

Ο αλγόριθμος ολοκληρώνεται σε n το πολύ επαναλήψεις (καθώς n είναι όλες οι ορθογώνιες κατευθύνσεις στον χώρο $\Re^n$). Κάθε επανάληψη έχεο κόστος $O(n^2)$ για πυκνούς πίνκαες και $O(n)$ για αραιούς πίνακες.

Στην πράξη, ο αλγόριθμος συγκλίνει σε πολύ λιγότερες από n επαναλήψεις. Αν ορισουμε την ποσότητα $\vec{r}^{(i)} = \vec{b} - A\vec{x}^{(i)}$ ως το υπόλοιπο ($residual$) κατά την i-οστή επανάληψη τότε το κριτήριο σύγκλισης της cg είναι:

$$\frac{\|\vec{r}^{(i)}\|}{\|\vec{b}\|} \leq itol \Leftrightarrow$$

$$\frac{\|\vec{b} - A\vec{x}^{(i)}\|}{\|\vec{b}\|} \leq itol$$

Η σχέση αυτή εκφράζει πως το σχετικό μήκος του υπολοίπου ως προς το μήκος του διανύσματος δεξιού μέλους πρέπει να είναι μικρότερο από ένα προκαθορισμενο κατώφλι σύγκλισης ($threshold$). Συνήθως, αυτό το κατώφλι σύγκλισης ορίζεται από τον χρήστη σε μία τιμή 10^{-3} έως 10^{-6}. Για το μήκος ($norm$) χρησιμοποιείται το Ευκλείδειο μήκος ($norm - 2$):

$$\|\vec{r}^{(i)}\| = \|\vec{r}^{(i)}\|_2 = \sqrt{\vec{r}^{(i)^T}\vec{r}^{(i)}}$$

Ως αρχική προσέγγιση χρησιμοποιούμε το $\vec{x}^{(0)} = [0, 0, ..., 0]^T$ αν δεν έχουμε κάποια άλλη πληροφορία για το σύστημά μας. Παρακάτω δίδεται ο αλγόριθμος της $conjugate\ gradient$:

Algorithm 5 CongugateGradient

Data: A, $\vec{b}$, $\vec{x}^{(0)}$, P
Result: $\vec{x}$
$\vec{x} = \vec{x}^{(0)}$ /* initial guess */
$\vec{r} = \vec{b} - A\vec{x}$ /* initial residual */
iter = 0
while $(\frac{\|\vec{r}\|}{\|\vec{b}\|} > itol)$ *and* $(iter < n)$ **do**
  iter = iter + 1
  SOLVE $M\vec{z} = \vec{r}$
  rho = $\vec{r}^T \cdot \vec{z}$
  if *iter == 1* **then**
    $\vec{p} = \vec{z}$
  else
    $beta = \frac{rho}{rho1}$
    $\vec{p} = \vec{z} + beta * \vec{p}$
  end
  rho1 = rho
  $\vec{q} = A\vec{p}$
  $alpha = \frac{rho}{\vec{p}^T \cdot \vec{q}}$
  $\vec{x} = \vec{x} + alpha * \vec{p}$
  $\vec{r} = \vec{r} - alpha * \vec{q}$
end

Θα αναλύσουμε θέματα που αφορούν την ταχύτητα σύγκλισης της cg. Για δύο διαδοχικές προσεγγίσεις της λύσης $\vec{x}^*$, $\vec{x}^{(i-1)}$ και $\vec{x}^{(i)}$ ισχύει οτι:

$$\frac{\|\vec{x}^{(i)} - \vec{x}^*\|_A}{\|\vec{x}^{(i-1)} - \vec{x}^*\|_A} = \frac{\sqrt{\lambda_{max}(A)} - \sqrt{\lambda_{min}(A)}}{\sqrt{\lambda_{max}(A)} + \sqrt{\lambda_{min}(A)}}$$

όπου $\|\vec{y}\|_A = \sqrt{\vec{y}^T A\vec{y}}$ είναι το A-μήκος $(A - norm)$ ή νόρμα ενέργειας ενός διανύσματος $\vec{y}$.

Οι τιμές $\lambda_{max}(A)$ και $\lambda_{min}(A)$ είναι αντίστοιχα, η μέγιστη και η ελάχιστη ιδιοτιμή του πίνακα A. Όταν ο πίνακας A είναι SPD έχει πραγματικές και θετικές ιδιοτιμές (πραγματικές επειδή είναι συμμετρικός και θετικές επειδή είναι θετικά ορισμένος).

Ο (φασματικός) δείκτης κατάστασης $(condition\ number)$ του πίνακα A ορίζεται ως:

$$\kappa_2(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\sqrt{\lambda_{max}(A^T A)}}{\sqrt{\lambda_{min}(A^T A)}}$$

Οι ιδιτιμές είναι του πίνακα $A^T A$ ονομάζονται ιδιάζουσες τιμές $(singular\ values)$ του πίνακα A. Για τον φασματικό δείκτη κατάστασης ισχύει οτι: $\kappa_2(A) \geq 1$ και μάλιστα όταν ο πίνακας είναι SPD θα ισχύει:

$$\sqrt{\lambda_i(A^T A)} = \sqrt{\lambda_i(A^2)} = \sqrt{\lambda_i^2(A)} = |\lambda_i(A)| = \lambda_i(A)$$

και επομένως:

$$\kappa_2(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$$

Τότε η σχέση σύγκλισης της cg γράφεται ως:

$$\frac{\|\vec{x}^{(i)} - \vec{x}^*\|_A}{\|\vec{x}^{(i-1)} - \vec{x}^*\|_A} \leq \frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1}$$

όπου:

$$0 \leq \frac{\sqrt{\kappa_2(A)}-1}{\sqrt{\kappa_2(A)}+1} \leq 1$$

Άρα λοιπόν, η cg συγκλίνει γρήγορα όταν $\kappa_2(A) \approx 1$, δηλαδή ο πίνακας A βρίσκεται σε καλή κατάσταση ($well-conditioned\ A$), ενώ αντιθέτως, έχουμε αργή σύγκλιση όταν $\kappa_2(A) >> 1$, δηλαδή ο πίνακας A βρίσκεται σε κακή κατάσταση ($ill-conditioned\ A$).

Η κακή κατάσταση θεραπεύεται σε κάποιο βαθμό με την χρήση ενός προρυθμιστή κατάστασης ($preconditioner$). Ο $preconditioner$ είναι ένας πίνακας M που εφαρμόζεται ώστε να λύσει το ισοδύναμο σύστημα:

$$M^{-1}A\vec{x} = M^{-1}\vec{b}$$

Στην πράξη, αντί της εφαρμογής του πίνακα M^{-1} στο σύστημα $A\vec{x} = \vec{b}$, σε κάθε επανάληψη της cg επιλύεται ένα σύστημα του $preconditioner$ $M\vec{z} = \vec{r}$.

Ένας $preconditioner$ είναι αποδοτικός όταν:

1. $M \approx A$, ώστε $\kappa_2(M^{-1}A) \approx \kappa_2(I) = 1$.
2. Η επίλυση συστημάτων $M\vec{z} = \vec{r}$ είναι πολύ ευκολότερη της επίλυσης του συστήματος $A\vec{x} = \vec{b}$.

Ο πιο απλός $preconditioner$ γενικού σκοπού ($general-purpose\ preconditioner$) είναι η διαγώνιος του A, $M = diag(A)$ ο οποίος ονομάζεται $Jacobi$ ή $diagonal\ preconditioner$. Ο $Jacobi\ preconditioner$ είναι πολύ αποδοτικός κυρίως όταν ο πίνακας A έχει διαγώνια κυριαρχία ($diagonally\ dominant$), δηλαδή όταν:

$$|a_{ii}| = \sum_{i=1, j\neq i}^{n} |a_{ij}| \ , \ \forall\ i = 1, 2, .., n$$

Ο στόχος αυτής της εργασίας είναι η εξαγωγή αποδοτικών $preconditioners$ για πίνακες που συσχετίζονται με γράφους, όπως ακριβώς στο πρόβλημα της προσομοίωσης κυκλωμάτων. Για το σκοπό αυτό, θα εκμεταλλευτούμε τεχνικές από την θεωρία γράφων για την επιτάχυνση της σύγκλισης της cg. Στο επόμενο κεφάλαιο θα αναλύσουμε την θεωρία γράφων και πως συσχετίζεται με τους πίνακες που προκύπτουν από τα κυκλώματα.

3.2.2 Η μέθοδος $bi-cg$

Στην προηγούμενη υποπαράγραφο, μελετήσαμε την μέθοδο cg για την επίλυση γραμμικών συστημάτων $A\vec{x} = \vec{b}$, όπου ο A είναι πίνακας SPD. Θα δούμε τώρα την επέκταση του αλγορίθμου αυτού για γενικούς πίνακες. Η επέκταση του αλγορίθμου βασίζεται στην μέθοδο ορθογώνιων προβολών ή μέθοδος υποχώρων $Krylov$ για μη-συμμετρικούς πίνακες. Όπως είδαμε η προσέγγιση της λύσης σε κάθε επανάληψη είναι:

$$\vec{x}^{(i)} = \vec{x}^{(i-1)} + a_i\vec{\rho}^{(i)}$$

άρα και το υπόλοιπο μετακινείται ως:

$$\vec{r}^{(i)} = \vec{b} - A\vec{x}^{(i)} = \vec{b} - A\vec{x}^{(i-1)} - a_i\vec{\rho}^{(i)} = \vec{r}^{(i-1)} - a_i\vec{\rho}^{(i)}$$

και αντίστοιχα μετακινούνται και οι κατευθύνσεις αναζητήσεως ως:

$$\vec{\rho}^{(i)} = \vec{r}^{(i)} + \beta_i\vec{\rho}^{(i-1)}$$

Αναπτύσσοντας την αναδρομή προκύπτει:

$$\vec{x}^{(i)} = \vec{x}^{(0)} + c_0\vec{r}^{(0)} + c_1 A\vec{r}^{(0)} + c_2 A^2\vec{r}^{(0)} + ... \Rightarrow$$

$$\vec{x}^{(i)} = \vec{x}^{(0)} + \sum_{j=0}^{i-1} c_j A^j \vec{r}^{(0)}$$

όπου τα c_j είναι σταθερές. Προκύπτει δηλαδή οτι το $\vec{x}^{(i)}$ ανήκει στον υπόχωρο:

$$K_i(A, \vec{r}^{(0)}) = span\{\vec{r}^{(0)}, A\vec{r}^{(0)}, A^2\vec{r}^{(0)}, ..., A^{i-1}\vec{r}^{(0)}\}$$

Αυτός είναι ο υπόχωρος $Krylov$ διάστασης i. Γενικά για A μη συμμετρικό ο A^TA είναι SPD καθώς $\forall\ \vec{x}\ \epsilon\ \Re^n, \vec{x} \neq \vec{0}$ είναι $\vec{x}^T A^T A\vec{x} = (A\vec{x})^T A\vec{x} > 0$ (λόγω εσωτερικού γινομένου), οπότε η cg είναι εφαρμόσιμη στο σύστημα:

$$A^TA\vec{x} = A^T\vec{b}$$

Όμως ο δείκτης κατάστασης του πίνακας A^TA είναι:

$$\kappa_2(A^TA) = \frac{\lambda_{max}(A^TA)}{\lambda_{min}(A^TA)} = (\frac{\sqrt{\lambda_{max}(A^TA)}}{\sqrt{\lambda_{min}(A^TA)}})^2 = (\kappa_2(A))^2$$

οπότε η σύγκλιση είναι της cg στον A^TA είναι τετραγωνικά πιο αργή.

Η μέθοδος $bi-cg$ είναι της cg σε αλγοριθμικό επίπεδο. Δεν ελαχιστοποιεί κάποια συνάρτηση $f(\vec{x})$ και επομένως δεν συγκλίνει πάντα. Συγκεκριμένα, υπάρχουν δύο περιπτώσεις αποτυχίας του αλγορίθμου. Δημιουργεί δύο ακολουθίες διανυσμάτων $\vec{\rho}^{(i)}$ και $\vec{\rho}^{(i)}$ και αντίστοιχα δύο ακολουθίες υπολοίπων $(residuals)$ $\vec{r}^{(i)}$ και $\vec{r}^{(i)}$ τα οποία είναι αμοιβαίως ορθογώνια $(mutually\ A-orthogonal)$, δηλαδή:

$$\vec{\rho}^{(i)^T} A\vec{\rho}^{(i-1)} = \vec{\rho}^{(i)^T} A\vec{\rho}^{(i)} = 0$$

Παρακάτω δίδεται ο αλγόριθμος της μεθόδου $bi-cg$:

Algorithm 6 BiCongugateGradient

Result: Solution $\vec{x}$ of $A\vec{x} = \vec{b}$ system
$\vec{x} = \vec{x}^{(0)}$ /* initial guess */
$\vec{r}$ = $\tilde{\vec{r}}$ = $\vec{b}$ - $A\vec{x}$ /* initial residual */
iter = 0
while $(\frac{\|\vec{r}\|}{\|\vec{b}\|} > itol)$ *and* $(iter < n)$ **do**
  iter = iter + 1
  SOLVE $M\vec{z} = \vec{r}$
  SOLVE $M^T\tilde{\vec{z}}$ = $\tilde{\vec{r}}$
  $rho = \vec{z} \cdot \tilde{\vec{r}}^T$
  if |*rho*| < *EPS* **then**
    EXIT /* algorithm failure */
  end
  if *iter* == *1* **then**
    $\vec{p} = \vec{z}$
    $\tilde{\vec{p}} = \tilde{\vec{z}}$
  else
    $beta = \frac{rho}{rho1}$
    $\vec{p} = \vec{z} + beta * \vec{p}$
    $\tilde{\vec{p}} = \tilde{\vec{z}} + beta * \tilde{\vec{p}}$
  end
  rho1 = rho
  $\vec{q} = A\vec{p}$
  $\tilde{\vec{q}} = A\tilde{\vec{p}}$
  $omega = \tilde{\vec{p}}^T \cdot \vec{q}$
  if |*omega*| < *EPS* **then**
    EXIT /* algorithm failure */
  end
  $alpha = \frac{rho}{omega}$
  $\vec{x} = \vec{x} + alpha * \vec{p}$
  $\vec{r} = \vec{r} - alpha * \vec{q}$
  $\tilde{\vec{r}}$ = $\tilde{\vec{r}}$ - alpha*$\tilde{\vec{q}}$
end

Η τιμή EPS είναι η μέγιστη ακρίβεια που θέτουμε στις αριθμητιές πράξεις. Στην υλοποίηση μας χρησιμοποιούμε για το EPS την τιμή 10^{-15}.

Κεφάλαιο 4

Θεωρία γράφων και εξαγωγή αποδοτικών προρυθμιστών κατάστασης (*preconditioners*)

Στόχος αυτού του κεφαλαίου είναι να περιγράψουμε και να αναλύσουμε τους αλγορίθμους για την εξαγωγή αποδοτικών *preconditioners*, καθώς και το θεωρητικό υπόβαθρο που απαιτείται για την κατανόηση αυτών. Θα ξεκινήσουμε αναφέροντας μερικά βασικά στοιχεία από την αλγεβρική θεωρία γράφων.

4.1 Εισαγωγή στους γράφους και στους Λαπλασιανούς πίνακες

Θεωρούμε έναν απλό, βεβαρημένο, μη-κατευθυνόμενο γράφο $G = (V, E, w)$, όπου V το σύνολο κορυφών (*vertices*) με $|V| = n$ και E το σύνολο των ακμών (*edges*) με $|E| = m$ του γράφου. Η συνάρτηση $w : E \to \Re_{>0}$ αναθέτει τα βάρη (*weights*) (ή μήκη -*lengths*-) των ακμών, τα οποία είναι πραγματικοί και θετικοί αριθμοί. Δηλώνουμε με $w(u, v)$ την ακμή μεταξύ των κορυφών u και v βάρους w. Ο βαθμός (*degree*) μιας κορυφής $v \epsilon V$ ορίζεται ως το άθροισμα των βαρών των ακμών που προσπίπτουν στην κορυφή v, δηλαδή:

$$deg(v) = \sum_{(v,u)\epsilon E} w(v, u)$$

Ορίζουμε για τον γράφο $G = (V, E, w)$ τον $n \times n$ πίνακα βαθμών κορυφών (*degree matrix*) D_G, ο οποίος περιέχει την πληροφορία για τους βαθμούς όλων των κορυφών του γράφου G. Ο πίνακας D_G ορίζεται ως:

$$d_{u,v} = \begin{cases} deg(v), \ if \ u = v \\ 0, \ otherwise \end{cases}$$

Αντίστοιχα, ορίζεται ο $n \times n$ πίνακας γειτνίασης (*adjacency matrix*) A_G, ο οποίος περιέχει την πληροφορία για το ποιοί κόμβοι συνδέονται (γειτνιάζουν). Ο πίνακας A_G ορίζεται ως:

$$a_{u,v} = \begin{cases} w(u, v), \ if \ (u, v)\epsilon E \\ 0, \ otherwise \end{cases}$$

Τέλος, ορίζεται ο Λαπλασιανός πίνακας (*Laplacian matrix*) του γράφου ως:

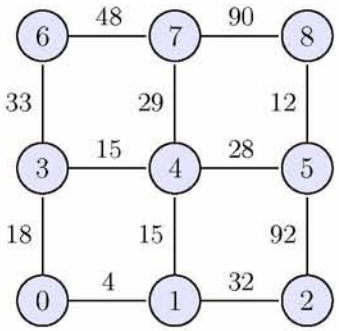
$$L_G = D_G - A_G$$

Προφανώς, ο *Laplacian* πίνακας L_G του γράφου G είναι πίνακας τέτοιος ώστε:

$$L_G(u,v) = \begin{cases} -w(u,v), \ if \ (u,v) \epsilon E \\ deg(v), \ if \ u = v \\ 0, \ otherwise \end{cases}$$

Ας δούμε όμως τους πίνακες αυτούς με ένα παράδειγμα.

- Έστω ο γράφος $G = (V, E, w)$ όπου το σύνολο κορυφών του είναι το $V = \{0,1,2,3,4,5,6,7,8\}$, το σύνολο ακμών του είναι το $E = \{(0,1),(1,2),(0,3),(1,4),(2,5),(3,4),(4,5),(3,6),(4,7),(5,8),(6,7),(7,8)\}$ και η συνάρτηση $w : E \to \Re_{>0}$ η οποία αναθέτει τα βάρη των ακμών όπως φαίνεται στο σχήμα.



- Επίσης, ο Λαπλασιανός πίνακα L_G του γράφου G, θα είναι ο ακόλουθος.

$$L_G = \begin{bmatrix} 22 & -4 & & -18 & & & & & \\ -4 & 51 & -32 & & -15 & & & & \\ & -32 & 124 & & & -92 & & & \\ -18 & & & 66 & -15 & & -33 & & \\ & -15 & & -15 & 87 & -28 & & -29 & \\ & & -92 & & -28 & 132 & & & -12 \\ & & & -33 & & & 81 & -48 & \\ & & & & -29 & & -48 & 167 & -90 \\ & & & & & -12 & & -90 & 102 \end{bmatrix}$$

Βέβαια, ο *Laplacian* πίνακας L_G του γράφου G ορίζεται και από την τετραγωνική μορφή (*quadratic form*) την οποία προκαλεί για οποιοδήποτε διάνυσμα $\vec{x} \ \epsilon \Re^{|V|} \Leftrightarrow \vec{x} \ \epsilon \Re^n$:

$$\vec{x}^T L_G \vec{x} = \sum_{w(u,v) \epsilon E} w(\vec{x}(u) - \vec{x}(v))^2$$

Ακολουθούν μερικές ιδιότητες του $n \times n$ *Laplacian* πίνακα L_G, με ιδιοτιμές $\lambda_0 \leq \lambda_1 \leq ... \leq \lambda_{n-1}$, ενός μη-κατευθυνόμενου γράφου G:

- Ο L_G είναι συμμετρικός ($L_G = L_G^T$) και έχει διαγώνια κυριαρχία ($|L_G(i,i)| = \sum_{i=1,j\neq i}^{n} |L_G(i,j)|$, $\forall\, i = 1,2,..,n$). Τέτοιοι πίνακες καλούνται, για συντομογραφία, SDD πίνακες (*Symmetric and Diagonally Dominant*).

- Ο L_G είναι θετικά ημιορισμένος και επομένως $\vec{x}TL_G\vec{x} \geq 0, \forall \vec{x} \epsilon \Re^n$ καθώς και $\lambda_i \geq 0 \forall i$.

- Ο L_G είναι M-πίνακας ($M - matrix$) (Στην γενική περίπτωση αυτό σημαίναει, οτι παρ'όλο που τα στοιχεία εκτός διαγωνίου ($off - diagonal\ elements$) δεν είναι θετικά, τα πραγματικά μέρη των ιδιοτιμών του, δεν είναι αρνητικά).

- Το άθροισμα κάθε γραμμής ($row\ sum$) και κάθε στήλης ($column\ sum$) του L_G είναι ίσο με το μηδέν.

- Κατά συνέπεια, $\lambda_0 = 0$, καθώς το διάνυσμα $\vec{v}_0 = [1,1,...,1]^T$ ικανοποιεί την σχέση $L_G\vec{v}_0 = \vec{0}$.

- Ο αριθμός εμφανίσεων του 0 ως ιδιοτιμή του πίνακα L_G είναι και ο αριθμός των συνδεδεμένων μερών ($connected\ components$) του γράφου.

- Η μικρότερη μη-μηδενική ιδιοτιμή του L_G ονομάζεται φασματική διαφορά ($spectral\ gap$).

- Η δεύτερη μικρότερη ιδιοτιμή του L_G ονομάζεται αλγεβρική συνδεσιμότητα ($algebraic\ connectivity$) (ή τιμή $Fiedler$). Το μέτρο της τιμής αυτής αντιπροσωπεύει πόσο καλά είναι συνδεδεμένος ο συνολικός γράφος.

- Ο $Laplacian$ πίνακας L_G αποτελεί ουσιαστικά ένα τελεστή Λαπλάς ($Laplace\ operator$) επί του n-διάστατου διανυσματικού χώρου των συναρτήσεων $f : V \to \Re$, όπου V είναι το σύνολο κορυφών του γράφου G και $n = |V|$.

- Για ένα γράφο G με πολλαπλά συνδεδεμένα μέρη ($connected\ components$) ο L_G είναι μπλοκ διαγώνιος πίνακας ($block\ diagonal\ matrix$), όπου το κάθε $block$ είναι ο αντίστοιχος $Laplacian$ πίνακας του ξεχωριστού υπογράφου (ίσως μετά από επαναδιάταξη των κορυφών).

- Ο κανονικοποιημένος Λαπλασιανός πίνακας ($normalized\ Laplacian$) ορίζεται ως $D^{1/2}L_GD^{1/2}$ και είναι πιο στενά συνδεδεμένος με την συνπεριφορά των τυχαίων περιπάτων ($random\ walks$) επί του γράφου G.

- Για δύο γράφους G και H με το ίδιο σύνολο κορυφών, ο γράφος $G + H$ ορίζεται ως ο γράφος που ο $Laplacian$ πίνακάς του είναι ο $L_G + L_H$.

Επομένως, υπάρχει ένας ισομορφισμός του $Laplacian$ πίνακα L_G και του αντίστοιχου γράφου G. Μπορούμε επομένως, να συσχετίσουμε έναν οποιδήποτε $n \times n$ $Laplacian$ πίνακα L_G, με ένα γράφο $G = (V, E)$, όπου το σύνολο κορυφών του θα είναι το $V = 1,2,...,n$ και θα έχει ακμές ανάμεσα στις κορυφές u και v βάρους $-L_G(u,v)$, για κάθε u, v με $A(u,v)$ μη-μηδενικό. Επίσης, παρατηρούμε οτι ο L_G ενός γράφου G δεν είναι αντιστρέψιμος ($singular\ matrix$).

Στην περίπτωση των ηλεκτρικών κυκλωμάτων είδαμε όταν επιχειρούμε να προσομοιώσουμε ένα ηλεκτρικό κύκλωμα με στοιχεία της ομάδας 1 (αντιστάτες, πυκνωτές και πηγές ρεύματος) καταλήγουμε στο γραμμικό σύστημα:

$$A_1GA_1^T\vec{v} = -A_1\vec{s_1} \Leftrightarrow \tilde{G}\vec{x} = \vec{b}$$

Ο πίνακας $\tilde{G} = A_1 G A_1^T$ είναι ο *Laplacian* πίνακας του γράφου που ορίζουν οι αντιστάτες του κυκώματος. Καθώς ο *Laplacian* πίνακας περιέχει τις τιμές των αγωγιμοτήτων των αντιστατών, ο αντίστοιχος γράφος θα έχει ως βάρος ακμών τις αντίστοιχες αγωγιμότητες. Ωστόσο, ο πίνακας G είναι αντιστρέψιμος, καθώς οι αντιστάτες που το ένα άκρο τους συνδέεται στον κόμβο αναφοράς (γείωση) συνεισφέρουν μόνο στο αντίστοιχο διαγώνιο στοιχείο και επομένως το άθροισμα κάποιων γραμμών (και στηλών) δεν θα είναι πλέον μηδεν.

Μετά από τις παρατηρήσεις αυτές, ας δούμε πως μπορούμε να επιταχύνουμε τις επαναληπτικές μεθόδους επίλυσης γραμμικών συστημάτων που αφορούν *Laplacian* πίνακες. Όπως είδαμε, η επιτάχυνση των μεθόδων αυτών μπορεί να επιτευχθεί μέσω της χρήσης αποδοτικών προρυθμιστών κατάστασης (*preconditioners*) B, οι οποίοι αποτελούν αφένός μια πολύ καλή προσέγγιση του αρχικού πίνακα A και αφέτέρου η επίλυση γραμμικών συστημάτων του B να είναι εύκολη.

Για παράδειγμα, κάθε επανάληψη της *Preconditioned Conjugate Gradient* (PCG) επιλύει ένα γραμμικό σύστημα του πίνακα B και πολλαπλασιάζει ένα διάνυσμα με τον πίνακα A. Ο αριθμός των επαναλήψεων που χρειάζεται η PCG για να βρεί μια ε-προσέγγιση ($\varepsilon - approximation$) της λύσης ενός γραμμικού συστήματος του A από τον σχετικό δείκτη κατάστασης (*relative condition number*) του A σε σχέση με τον B, $\kappa(A, B)$. Για συμμετρικούς και θετικά ορισμένους πίνακες A και B ο σχετικός δείκτης κατάστασης ορίζεται ως ο λόγος της μεγαλύτερης προς την μικρότερη ιδιοτημή του AB^{-1}:

$$\kappa(A, B) = \frac{\lambda_{max}(AB^{-1})}{\lambda_{min}(AB^{-1})}$$

Μπορεί να δειχθεί οτι η PCG θα βρεί μία ε-προσέγγιση της λύσης το πολύ $O(\sqrt{\kappa(A,b)} \log \varepsilon^{-1})$ επαναλήψεις. Επομένως οι αποδοτικοί *preconditioners* αποδεικνύονται πάρα πολύ χρήσιμοι στην πράξη

4.2 Προρυθμιστές κατάστασης από υπογράφους και θεωρία στήριξης

Η επαναστατική ιδέα που έδωσε ώθηση για έρευνα, ήταν η ιδέα του *Pravin Vaidya* να κατασκευάσει *preconditioners* για *Laplacian* πίνακες οι οποίοι προκύπτουν από γράφους, από τους *Laplacian* πίνακες υπογράφων (*subgraphs*) των γράφων αυτών. Εκ τότε, έχει γίνει μεγάλη έρευνα προς αυτήν την κατεύθυνση. Η οικογένεια *preconditioners* που προέκυψαν αναφέρονται ως προρυθμιστές κατάστασης υπογράφων (*subgraph preconditioner*) ή συνδιαστικοί προρυθμιστές κατάστασης (*combinatorial preconditioners*).

Ο κύριος σκοπός της εργασίας αυτής είναι να εξετάσουμε τους αλγορίθμους κατασκευής τέτοιων *preconditioners*, να τους εφαρμόσουμε στην επίλυση γραμμικών συστημάτων που προκύπτουν από τις ανάγκες της προσομοίωσης κυκλωμάτων και κατόπιν να συγκρίνουμε τα αποτελέσματα. Στις επόμενες παραγράφους θα μελετήσουμε τους προρυθμιστές κατάστασης υπογράφων που προκύπτουν από το μέγιστο επικαλύπτον δέντρο, το επικαλύπτον δέντρο χαμηλής έκτασης και το επαυξημένο επικαλύπτον δέντρο χαμηλής έκτασης.

Το θεωρητικό υπόβαθρο που χρησιμοποιείται για την ανάλυση αυτών καλείται 'θεωρία στήριξης' (*support theory*). Η θεωρία στήριξης χρησιμοποιεί συνδυαστικές τεχνικές για να αποδείξει ανισότητες Λαπλασιανών πινάκων που προκύπτουν από γράφους. Ο απώτερος σκοπός της διαδικασίας αυτής είναι να αποδειχθεί ένα φράγμα στον αριθμό των επαναλήψεων της PCG. Όπως είδαμε όταν επιλύουμε ένα σύστημα του πίνακα A, με την μέθοδο PCG, χρησιμοποιώντας ως *preconditioner*

τον πίνακα B, ο αριθμός των επαναλήψεων είναι ανάλογος του γενικευμένου δείκτη κατάστασης, δηλαδή της τετραγωνικής ρίζας του λόγου των ακραίων πεπερασμένων γενικευμένων ιδιοτιμών των πινάκων (A, B):

$$\kappa(A, B) = \frac{\lambda_{max}(AB^{-1})}{\lambda_{min}(AB^{-1})}$$

Σύμφωνα με τον ορισμό, ένας αριθμός $lambda$ είναι πεπερασμένη γενικευμένη ιδιοτιμή των πινάκων (A, B) αν υπάρχει ένα μη-μηδενικό διάνυσμα $\vec{x} \neq \vec{0}$ τέτοιο ώστε $A\vec{x} = \vec{x}$ και $B\vec{x} \neq \vec{0}$. Το σύνολο των πεπερασμένων ιδιοτιμών των πινάκων (A, B) συμβολίζεται ως $\lambda_f(A, B)$.

Άρα λοιπόν, για να βρεθεί ένα φράγμα στον αριθμό επαναλήψεων της PCG, πρέπει να βρεθούν φράγματα για τις ακραίες τιμές του συνόλου πεπερασμένων γενικευμένων ιδιοτιμών $\lambda_f(A, B)$. Πρέπει, δηλαδή, να βρεθεί ένα άνω φράγμα του $max\lambda_f(A, B)$ και ένα κάτω φράγμα για το $min\lambda_f(A, B)$.

Σύμφωνα λοιπόν με την θεωρία στήριξης, αν A και B είναι θετικά ημι-ορισμένοι πίνακες, τότε γράφουμε:

$$A \geq B$$

αν ο πίνακας $A - B$ είναι θετικά ημι-ορισμένος, δηλαδή αν για $\vec{x} \epsilon \Re^{|V|}$ ισχύει:

$$\vec{x}^T A\vec{x} \geq \vec{x}^T B\vec{x}$$

Έχει αποδειχθεί οτι αν $\sigma_{A,B}$ και $\sigma_{B,A}$ είναι οι ελάχιστες σταθερές για τις οποίες ισχύει:

$$\sigma_{A,B}A \geq B \;\; \sigma_{B,A}B \geq A,$$

τότε θα ισχύει ότι η ελάχιστη πεπερασμένη γενικευμένη ιδιοτιμή των A, B θα είναι:

$$min\lambda_f(A, B) = \lambda_{min}(AB^+) = \sigma_{A,B},$$

αντίστοιχα, η μέγιστη πεπερασμένη γενικευμένη ιδιοτιμή των A, B θα είναι:

$$max\lambda_f(A, B) = \lambda_{max}(AB^+) = \sigma_{B,A},$$

και τέλος, ο γενικευμένος δείκτης κατάστασης των πινάκων (A, B) θα είναι:

$$\kappa(A, B) = \sigma_{A,B}\sigma_{B,A}.$$

Τέτοιες ανισότητες αποδεικνύονται από την θεωρία στήριξης.

Ας θεωρήσουμε τώρα, ένα γράφο $G = (V, E, w)$ και έστω $H = (V, F, w)$ ένας υπογράφος αυτού. Γράφουμε w και στους δύο για να δηλώσουμε οτι οι ακμές που εμφανίζονται και στον G και στον H έχουν το ίδιο βάρος. Αν θεωρήσουμε επίσης τους $Laplacian$ πίνακες L_G και L_H των γράφων αυτών. Προκύπτει επομένως ότι:

$$\vec{x}^T L_G\vec{x} = \sum_{(u,v)\epsilon E} w_{u,v}(\vec{x}(u) - \vec{x}(v))^2 \geq \sum_{(u,v)\epsilon F} w_{u,v}(\vec{x}(u) - \vec{x}(v))^2 = \vec{x}^T L_H\vec{x}$$

και επομένως θα ισχύει:

$$L_G \geq L_H.$$

Τέτοιου είδους ανισότητες είναι φυσικές για τους Λαπλασιανούς πίνακες και για αυτό δημιουργούμε προρυθμιστές κατάστασης από υπογράφους. Πρωτού όμως προχωρήσουμε στους αλγορίθμους εξαγωγής των υπογράφων θα εξετάσουμε δομές δεδομένων αποθήκευσης των γράφων.

4.3 Δομές δεδομένων αναπαράστασης γράφων

Οι αλγόριθμοι που θα μελετήσουμε αφορούν το γράφο του ηλεκτρικού κυκλώματος και επομένως χρειαζόμαστε μια κατάλληλη δομή δεδομένων για την αναπαράσταση αυτού. Στην πράξη, οι βασικότερες δομές δεδομένων που χρησιμοποιούνται για την αναπαράσταση ενός γράφου $G = (V, E)$ είναι οι ακόλουθες:

Λίστα γειτνείασης (*adjacency list*)
Σε αυτήν την δομή χρησιμοποιείται ένας τύπος δεδομένων για την αναπαράσταση των κορυφών. Κάθε κόμβος κρατά μια λίστα με τις γειτονικές κορυφές του. Αυτή η δομή την αποθήκευση επιπλέον δεδομένων σχετικά με τις κορυφές (π.χ. τον βαθμό).

Πίνακας γειτνίασης (*adjacency matrix*)
Σε αυτήν την δομή χρησιμοποιείται ένας δυδιάστατος $|V| \times |V|$ πίνακας, όπου οι γραμμές αναπαριστούν κορυφές προέλευσης (*source vertices*) και οι στήλες κορυφές προορισμού (*destination vertices*). Η μόνη πληροφορία που μπορούμε να αποθηκεύσουμε στον πίνακα γειτνίασης είναι τα βάρη των ακμών μεταξύ γειτονικών κορυφών. Οποιαδήποτε άλλη πληροφορία σχετικά με τις κορυφές ή τις ακμές του γράφου πρέπει να αποθηκευτεί ξεχωριστά.

Πίνακας πρόσπτωσης (*incidence matrix*)
Σε αυτήν την δομή χρησιμοποιείται ένας δυδιάστατος $|V| \times |E|$ πίνακας, όπου οι γραμμές αναπαριστούν τις κορυφές και οι στήλες αναπαριστούν τις ακμές. Τα στοιχεία του πίνακα μας δείχνουν η κορυφή της γραμμής προσπίπτει στην ακμή της στήλης.

Στον παρακάτω πίνακα βλέπουμε την χρονική πολυπλοκότητα για να επιτελέσουμε διάφορες πράξεις επί των δομών αυτών:

operation	*adjacency list*	*adjacency matrix*	*incidence matrix*
αποθήκευση	$O(\|V\| + \|E\|)$	$O(\|V\|^2)$	$O(\|V\|\|E\|)$
προσθήκη κορυφής	$O(1)$	$O(\|V\|^2)$	$O(\|V\|\|E\|)$
προσθήκη ακμής	$O(1)$	$O(1)$	$O(\|V\|\|E\|)$
διαγραφή κορυφής	$O(E)$	$O(\|V\|^2)$	$O(\|V\|\|E\|)$
διαγραφή ακμής	$O(E)$	$O(1)$	$O(\|V\|\|E\|)$
αναζήτηση γειτονικών κορυφών	$O(\|V\|)$	$O(1)$	$O(\|E\|)$

Να σημειώσουμε οτι η λίστα γειτνίασης για την διαγραφή ακμών ή κορυφών απαιτεί όλων των ακμών ή κορυφών. Ο πίνακας γειτνίασης στην προσθήκη και διαγραφή κορυφών καθώς πρέπει να αλλάξει το μέγεθος του πίνακα. Κάτι αντίστοιχο συμβαίνει και στον πίνακα πρόσπτωσης οπού το μέγεθος του πίνακα πρέπει να αλλάξει είτε με την προσθήκη/διαγραφή κορυφών είτε με την προσθήκη/διαγραφή ακμών.

Ο πίνακας γειτνίασης χρησιμοποιείται κυρίως όταν ο γράφος είναι πυκνός, δηλαδή το πλήθος των ακμών $|E|$ του είναι κοντά στο τετράγωνο του πλήθους των κορυφών $|V|^2$ ή όταν απαιτείται γρήγορη εύρεση ακμής που συνδέει δύο κορυφές.

Ωστόσο, στην υλοποίησή μας χρησιμοποιούμε λίστα γειτνίασης για την αναπαράσταση του γράφου. Η λίστα γειτνίασης αναπαριστά πολύ αποδοτικά αραιούς γράφους (*sparse graphs*), πράγμα το οποίο μας είναι ιδιαίτερα χρήσιμο, καθώς στα κυκλώματα μεγάλης κλίμακας προκύπτουν αραιοί γράφοι καθώς αντίστοιχα και αραοί πίνακες.

4.4 Ελάχιστο επικαλύπτον δέντρο

Όπως αναφέραμε στην παράγραφο (4.2) ο *Vaidya* πρότεινε να κατασκευαστουν προρυθμιστές κατάστασης υπογράφων και πιο συγκεκριμένα προρυθμιστές κατάστασης επικαλύπτοντων δέντρων. Δηλαδή ως *preconditioner* ενός *Laplacian* ενός γράφου να χρησιμοποιηθεί ο *Laplacian* ενός επικαλύπτοντος δέντρου του γράφου αυτού.

Ως δέντρο είναι ένας συνδεδεμένος, μη-κατευθυνόμενος γράφος, ο οποίος δεν περιέχει κύκλους. Το επικαλύπτον δέντρο ενός γράφου G πρέπει να επικαλύπτει τον G, δηλαδή να περιέχει όλες τις κορυφές του G, και να είναι υπογράφος του G, δηλαδή κάθε ακμή του δέντρου να ανήκει στον γράφο G. Ένα επικαλύπτον δέντρο του συνδεδεμένου γράφου G μπορεί να οριστεί ως το μέγιστο σύνολο ακμών του G το οποίο δεν σχηματίζει κύκλο ή το ελάχιστο σύνολο ακμών που συνδέει όλες τις ακμές.

Καθώς μπορούμε να χρησιμοποιήσουμε μία άμεση (*direct*) μέθοδο για να λύσουμε το γραμμικό σύστημα του Λαπλασιανού πίνακα ενός επικαλύπτοντος δέντρου σε γραμμικό χρόνο, κάθε επανάληψη της PCG με *preconditioner* ένα επικαλύπτον δέντρο θα χρειαζόταν χρόνο $O(m+n)$ όπου m είναι ο αριθμός των ακμών του αρχικού γράφου.

Πιο συγκεκριμένα ο *Vaidya* πρότεινε ως *preconditioner* τον *Laplacian* του μέγιστου επικαλύπτοντος δέντρου (*maximum spanning tree*). Έχει αποδειχθεί οτι αν T είναι ένα μέγιστο επικαλύπτον δέντρο του γράφου G τότε: $(nm)L_T \geq L_G$.

Ο πίνακας που εξάγεται για την προσομοίωση του κυκλώματος περιέχει ως στοιχεία ηλεκτρικές αγωγιμότητες. Εν τούτοις, ο γράφος του κυκλώματος σχηματίζεται με βάση τις αντιστάσεις και όχι τις αγωγιμότητες. Όμως τα δύο αυτά μεγέθη είναι αντίστροφα και επομένως θα εξάγουμε το ελάχιστο επικαλύπτον δέντρο (*minimum spanning tree* $-MST-$) σε σχέση με τις αντιστάσεις.

Για τον υπολογισμό του ελάχιστου επικαλύπτοντος δέντρου χρησιμοποιούμε τον αλγόριθμο του *Prim*. Η γενική ιδέα του αλγορίθμου έχει ως εξής:

1. Επέλεξε αυθαίρετα μια κορυφή του γράφου και αρχικοποίησε το δέντρο με αυτήν.

2. Αύξησε το δέντρο κατά μία κορυφή: από τις ακμές που συνδέουν το δέντρο με κορυφές που δεν ανήκουν ακόμα στο δέντρο διάλεξε αυτή με το ελάχιστο βάρος και πρόσθεσέ την στο δέντρο.

3. Επανέλαβε το βήμα 2 μέχρι όλες οι κορυφές να ανήκουν στο δέντρο.

Μπορούμε να δούμε τον αλγόριθμο του *Prim* με περισσότερες λεπτομέρειες παρακάτω. Ο αλγόριθμος συσχετίζει κάθε κορυφή v του γράφου με την ακμή $E[v]$, η οποία φέρει το ελάχιστο βάρος και με ένα αριθμό $C[v]$ ο οποίος δηλώνει αυτό το ελάχιστο βάρος. Η αρξικοποίηση αυτών των τιμών γίνεται με $+\infty$ (ή DBL_MAX σε προγραμματιστικό επίπεδο) για τους αριθμούς $C[v]$ και με μία *boolean* μεταβλητή (*flag value*) για τις ακμές $E[v]$ η οποία θα δηλώνει οτι δεν υπάρχει ακμή που να συνδέει την κορυφή v με κορυφές που έχουν ήδη εξεταστεί και επομένως μπορεί να αρχικοποιηθεί σε $undefined$.

Επίσης, γίνεται αρχικοποίηση ενός άδειου δέντρου (*tree*) T και αρχικοποίηση του συνόλου Q των κορυφών που δεν περιλαμβάνονται στο T (αρχικά, όλες οι κορυφές).

Algorithm 7 *PrimAlgorithm*

Data: G=(V,E,w)
Result: T
/* initialization */
for $v \epsilon V$ **do**
  $C[v] = +\infty$
  $E[v] = undefined$
end
$T = \emptyset$
$Q = V$
/ main process */*
while $Q \neq \emptyset$ **do**
  $v = min_u\{C[u]\}$
  $Q = Q - \{v\}$
  $T = T \cup \{v\}$
  if $E[v] \neq undefined$ **then**
    $T = T \cup \{E[v]\}$
  end
  for $u \epsilon V : (u, v) \epsilon E$ **do**
    if $u \epsilon Q \ \ \& \ \ w(u, v) < C[u]$ **then**
      $C[u] = w(u, v)$
      $E[u] = (u, v)$
    end
  end
end
return T

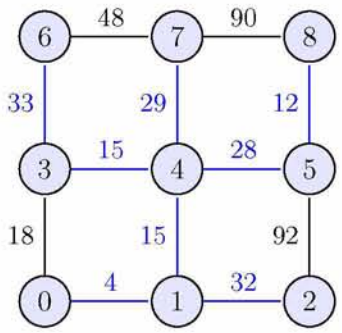
Όπως λοιπόν περιγράφεται παραπάνω, η αρχική κορυφή για τον αλγόριθμο επιλέγεται τυχαία. Στην υλοποίηση μας θα επιλέγει ο χρήστης την κορυφή από την οποία επιθυμεί να ξεκινήσει η διαδικασία και η επιλογή αυτή θα είναι μια παράμετρος στην συνάρτηση η οποία αναλαμβάνει να υπολογίσει το μέγιστο επικαλύπτον δέντρο. Όπως είδαμε προηγουμένως, οι κόμβοι του κυκλώματος, είναι οι κορυφές του αντίστοιχου γράφου και άρα ο χρήστης θα δίνει την επιλογή του στο αρχείο περιγραφής κυκλώματος ($netlist$).

Η υλοποίηση του αλγορίθμου βέβαια, εξαρτάται από τις δομές δεδομένων που έχουν χρησιμοποιούνται για να αναπαραστήσουν τον γράφο, το δέντρο, καθώς και το σύνολο Q που περιγράφει ο αλγόριθμος. Για την αναπαράσταση του γράφου χησιμοποιούμε λίστα γειτνίασης και καθώς το δέντρο είναι επίσης ένας γράφος θα χρησιμοποιήσουμε την ίδια δομή δεδομένων.

Εκτελώντας τον αλγόριθμο $Prim$ στον γράφο της παραγράφου 4.1, με αρχική κορυφή την κορυφή 4, ο αλγόριθμος θα δημιουργήσει το ελάχιστο επικαλύπτον δέντρο H προσθέτοντας διαδοχικά τις ακμές $(4, 3)$, $(4, 1)$, $(1, 0)$, $(4, 5)$, $(5, 8)$, $(4, 7)$, $(1, 2)$ και $(3, 6)$. Σημειώνουμε με μπλε χρώμα τις ακμές που ανήκουν στο ελάχιστο επικαλύπτον δέντρο στον γράφο της επόμενης εικόνας.

Δίνουμε επίσης τον Λαπλασιανό πίνακα αυτού του υπογράφου H ο οποίος θα είναι ο $preconditioner$ στην επίλυση του αντίστοιχου γραμμικού συστήματος. Βέβαια, στα πραγματικά προβλήματα τα οποία προσομοιώνουμε, ο Λαπλασιανός πίνακας του γράφου L_G του κυκλώματος και ο αντίστοιχος Λαπλασιανός του ελάχιστου επικαλύπτοντος δέντρου L_H θα περιέχουν τις αγωγιμότητες του κυκλώματος, παρ' όλο που ο γράφος του κυκλώματος κατασκευάζεται με βάση τις αντιστάσεις. Επίσης, στα πραγματικά προβλήματα θα υπάρχουν και αντιστάσεις προς την γείωση που θα καθιστούν και

τους δύο πίνακες αντιστρέψιμους.



$$L_H = \begin{bmatrix} 4 & -4 & & & & & & & \\ -4 & 51 & -32 & & -15 & & & & \\ & -32 & 32 & & & & & & \\ & & & 48 & -15 & & -33 & & \\ & -15 & & -15 & 87 & -28 & & -29 & \\ & & & & -28 & 40 & & & -12 \\ & & & -33 & & & 33 & & \\ & & & & -29 & & & 29 & \\ & & & & & -12 & & & 12 \end{bmatrix}$$

4.5 Επικαλύπτον δέντρο χαμηλής έκτασης

Στην παράγραφο αυτή θα μελατήσουμε τα απικαλύπτοντα δέντρα χαμηλής έκτασης και τους αλγορίθμους για την εξαγωγή τους. Οι *Bohman* και *Hendrickson* θεώρησαν οτι για την κατασκευή αποδοτικών προρυθμιστών κατάστασης (*preconditioners*) από επικαλύπτοντα δέντρα, πρέπει να μετρηθεί η έκταση (*stretch*) του δέντρου. Η ιδέα της έκτασης ενός επικαλύπτοντος δέντρου εισήχθηκε αρχικά από τους *Alon, Karp, Peleg* και *West* στην ανάλυση για το πρόβλημα των k-εξυπηρετητών ($k - server\ problem$). Ωστόσο, η έκταση των δέντρων μπορεί να οριστεί χωρίς καμία αναφορά στο πρόβλημα αυτό.

Ας θεωρήσουμε ένα βεβαρημένο, συνδεδεμένο γράφο $G = (V, E, w)$ όπου $w : E \to \Re_{>}0$ η συνάρτηση που αναθέτει τα βάρη (*weights*) ή μήκη (*lengths*) στις ακμές του γράφου. Δεδομένου ενός επικαλύπτοντος δέντρου T του G ορίζεται η απόσταση (*distance*) μεταξύ δύο κορυφών $u,\ v\ \epsilon\ V$ και συμβολίζεται ως $dist_T(u, v)$ να είναι το άθροισμα των βαρών/μηκών των ακμών στο μοναδικό μονοπάτι (*path*) στο T μεταξύ των κορυφών u και v. Έπειτα, ορίζεται η έκταση (*stretch*) μίας ακμής $(u, v)\ \epsilon\ E$ ως:

$$stretch_T(u, v) = \frac{dist_T(u, v)}{w(u, v)},$$

και η μέση έκταση *average stretch* επί όλων των ακμών του E ως:

$$ave - stretch_T(E) = \frac{1}{|E|} \sum_{(u,v)\epsilon E} stretch_T(u, v)$$

Έχει αποδειχθεί οτι κάθε βεβαρημένος, συνδεδεμένος γράφος $G = (V, E, w)$ με n κορυφές και m ακμές περιέχει ένα επικαλύπτον δέντρο τέτοιο ώστε:

$$ave - stretch_T(E) = \exp^{O(\sqrt{\log n \log\log n})}$$

Στην εργασία αυτή υλοποιούμε τον αλγόριθμο των *Elkin, Emek, Spielman* και *Teng* για την κατασκευή επικαλύπτοντων δέντρων χαμηλής έκτασης ($low - stretch\ spanning\ trees$). Ο αλγόριθμός τους κατασκευάζει ένα επικαλύπτον δέντρο $T \subset E$ για κάθε βεβαρημένο, συνδεδεμένο γράφο $G = (V, E, w)$ τέτοιο ώστε:

$$ave - stretch_T(E) = O(\log^2 nloglogn).$$

Το όριο αυτό είναι σημαντική βελτίωση σε σχέση με το $\exp^{O(\sqrt{\log n \log\log n})}$ το οποίο είδαμε νωρίτερα. Ο χρόνος εκτέλεσης του αλγορίθμου είναι $O(m \log n + nlog^2 n)$ για βεβαρημένους γράφους. Σημειώνεται οτι ο γράφος πρέπει να είναι απλός και ο αριθμός των ακμών του m να μην υπερβαίνει κατά πολύ το $\binom{n}{2}$, αν και το να θεωρούμε γράφους με το πολύ $n(n+1)$ ακμές είναι αρκετό. Οι γράφοι που προκύπτουν από την προσομοίωση κυκλωμάτων πληρούν αυτές τις προϋποθέσεις οπότε δεν θα μας απασχολήσουν περισσότερο.

Σημειώνουμε επίσης, οτι τα επικαλύπτοντα δέντρα χαμηλής εκτάσης χρησιμοποιούνται σε πολλές εφαρμογές και κάποιες από αυτές απαιτούν τον ορισμό της έκτασης μίας ακμής $(u, v)\ \epsilon\ E$ ως: $stretch_T(u, v) = \frac{dist_T(u,v)}{dist_G(u,v)}$. Αν και οι αλγόριθμοι που θα δούμε παρακάτω μπορούν να τροποποιηθούν με βάση τον ορισμό αυτό, στηρίζονται στον αρχικό ορισμό που δώσαμε. Οι κύριες εφαρμογές των επικαλύπτοντων δέντρων χαμηλής εκτάσης είναι στην επίλυση γραμμικών συστημάτων ($linear\ system\ solution$), στο πρόβλημα των k-εξυπηρετητών ($k - server\ problem$)και στη βελτίωση της προσέγγισης του προβλήματος επικαλύπτοντος δέντρου ελάχιστου επικοινωνιακού κόστους ($minimum\ communication\ cost\ spanning\ tree$). Προφανώς, το ενδιαφέρον μας επικεντρώνεται στην επίλυση γραμμικών συστημάτων καθώς οι *Spielman* και *Teng* απέδειξαν οτι για επίπεδα ($planar$) γραμμικά συστήματα ο χρόνος που απαιτείται για την επίλυσή τους είναι $O(n \log^2 n \log\log n \log \frac{1}{\varepsilon})$, όπου ε είναι η ακρίβεια της λύσης.

Ο τρόπος με τον οποίο κατασκευάζονται τα επικαλύπτοντα δέντρα χαμηλής εκτάσης είναι με της αναδρομικής εφραμογής μιας τεχνικής διάσπασης γράφων η οποία καλείται 'διάσπαση άστρου' ($star - decomposition$). Μία τέτοια διάσπαση του γράφου προκαλεί μια διαμέριση του συνόλου κορυφών V σε σύνολα που σενδέονται σε σχήμα άστρου ($star$): μια κεντρική κορυφή συνδέεται με όλα τα υπόλοιπα σύνολα μέσω μιας μοναδικής ακμής, όπως στην εικόνα. Ο αλγόριθμος για την $star - decomposition$ είναι έτσι σχεδιασμένος ώστε η ακτίνα των γράφων που παράγονται να μην είναι πολύ μεγαλύτερη από αυτήν του αρχικού γράφου.

Πρωτού αναλύσουμε τον αλγόριθμο για την κατασκευή του επικαλύπτοντος δέντρου χαμηλής έκτασης, ας δούμε μερικές βασικές σχετικές έννοιες. Θεωρούμε έναν βεβαρημένο, συνδεδεμένο, επίπεδο γράφο $G = (V, E, w)$ με πλήθος κορυφών $|V| = n$, πλήθος ακμών $|E| = m$ και $w : E \to \Re_{>0}$ η συνάρτηση που αναθέτει τα βάρη ($weights$) ή μήκη ($lengths$) στις ακμές. Το κόστος ($cost$) ή αντίσταση ($resistance$) μίας ακμής $(u, v)\ \epsilon\ E$ ορίζεται ως το αντίστοφο του βάρους της, δηλαδή:

$$cost(u, v) = \frac{1}{w(u, v)}.$$

Αντίστοιχα, το κόστος ενός συνόλου ακμών $F \subseteq E$ συμβολίζεται ως $cost(F)$ και ορίζεται ως το άθροισμα των κόστων των ακμών του F, δηλαδή:

$$cost(F) = \sum_{(u,v)\epsilon F} cost(u,v) = \sum_{(u,v)\epsilon F} \frac{1}{w(u,v)}$$

Θεωρώντας δύο οποιεσδήποτε κορυφές $u, v \epsilon V$ η απόσταση $dist(u,v)$ ορίζεται ως το μήκος (βάρος) του συντομότερου μονοπατιού μεταξύ των u και v στον γράφο G. Γράφεται και ως $dist_G(u,v)$ για να τονίστει οτι η απόσταση αφορά τον γράφο G. Για ένα υποσύνολο κορυφών $S \subseteq V$ ορίζεται η απόσταση μεταξύ της κορυφής u και του συνόλου S ως:

$$dist_G(u,S) = min\{dist_G(u,x) \mid x \epsilon S\}$$

Για ένα σύνολο κορυφών $S \subseteq V$, $G(S)$ είναι ο υπογράφος που προκύπτει στον G από τις κορυφές που ανήκουν στο S. Εφ'όσον αυτό είναι κατανοητό αντί για $dist_{G(S)}(u,v)$ η απόσταση στον υπογράφο αυτόν μπορεί να γραφεί και ως $dist_S(u,v)$, δηλαδή

$$dist_S(u,v) = dist_{G(S)}(u,v).$$

Επίσης, ως $E(S)$ ορίζεται το σύνολο των ακμών που και τα δύο του άκρα ανήκουν στο S. Το σύνορο (*boundary*) του S, συμβολίζεται ως $\partial(S)$ είναι το σύνολο των ακμών με μόνο ένα άκρο να ανήκει στο S.

$$\partial(S) = \{(u,v)\epsilon E : u \in S, v \notin S\}$$

Εάν T είναι ένα σύνολο κορυφών και $S \cap T = \emptyset$ τότε $E(S,T)$ είναι το σύνολο των ακμών με το ένα άκρο να ανήκει στο S και το άλλο άκρο να ανήκει στο T. Μία πολυμερής διαμέριση (*multiway partition*) του συνόλου V είναι μια συλλογή από ανά δύο ξένα μεταξύ τους σύνολα $\{V_1, V_2, ..., V_k\}$ τέτοια ώστε $\cup_i V_i = V$.

Το σύνορο (*boundary*) μίας πολυμερούς διαμέρισης, συμβολίζεται με $\partial(V_1, V_2, ..., V_k)$ και είναι το σύνολο των ακμών που έχουν τα άκρα τους σε διαφορετικά σύνολα της διαμέρισης. Ο όγκος (*volume*) ενός συνόλου ακμών F, συμβολίζεται με $vol(F)$ και είναι ο πληθάριθμος του συνόλου F, $|F|$. Αντίστοιχα, ο όγκος ενός συνόλου κορυφών S, συμβολίζεται με $vol(S)$ και είναι ο αριθμός των ακμών με τουλάχιστον το ένα άκρο να ανήκει στο S.

Για μία κορυφή $v \epsilon V$ η ακτίνα (*radius*) του G σε σχέση με την v, συμβολίζεται ως $rad_G(v)$ και είναι το ελάχιστο r για το οποίο κάθε κορυφή του G βρίσκεται σε απόσταση το πολύ r από την v. Εφ'όσον αυτό είναι κατανοητό, για ένα υποσύνολο κορυφών $S \subseteq V$ αντί για $rad_{G(S)}(v)$ η ακτίνα στον υπογράφο αυτόν μπορεί να γραφεί και ως $rad_S(v)$.

Ο υπολογισμός του $rad_G(v)$ απαιτεί την γνώση όλων των αποστάσεων των κορυφών του γράφου G από την κορυφή v. Έτσι, η ακτίνα του γράφου G σε σχέση με την κορυφή v θα είναι η μέγιστη απόσταση που μπορεί να έχει μια κορυφή u από την κορυφή v.

Ο τρόπος που μπορούμε να το υπολογίσουμε αυτό είναι μέσω του αλγορίθμου *Dijkstra*. Ο αλγόριθμος *Dijkstra* υπολογίζει τα συντομότερα μονοπάτια και κατ' επέκταση τις συνομότερες αποστάσεις από μία κορυφή v του γράφου, και άρα η μέγιστη εξ αυτών θα είναι και η ακτίνα του γράφου σε σχέση με την συγκεκριμένη κορυφή.

Ο αλγόριθμος ξεκινάει από μία κορυφή (*source*) στην οποία θέτει την απόσταση ίση με μηδέν και αρχικοποιεί όλες τις υπόλοιπες κορυφές με άπειρη απόσταση. Έπειτα, τοποθετεί όλες τις κορυφές που

πρέπει να εξετάσει σε ένα σύνολο Q. Σε κάθε επανάληψη επιλέγεται να εξετασθεί η κορυφή u με την μικρότερη απόσταση. Η κορυφή u αφαιρείται από το σύνολο Q και στην συνέχεια ελέγχεται αν μέσω αυτής της κορυφής u δημιουργούνται συντομότερα μονοπάτια προς τις γειτονικές τις κορυφές. Έτσι, θέτονται κατάλληλα οι τιμές των γειτόνων της u και ο αλγόριθμος περνά σε επόμενη επανάληψη. Στην χειρότερη περίπτωση η πολυπλοκότητα του αλγορίθμου είναι $O(|E| + |V| \log |V|)$.

Επίσης, ο αλγόριθμος αυτός μπορεί να τροποποιηθεί ώστε να εφαρμόζεται σε έναν υπογράφο, που ορίζεται από ένα υποσύνολο κορυφών του αρχικού γράφου. Στον διαδικασία εύρεσης του επικαλύπτοντος δέντρου χαμηλής έκτασης θα χρειαστούμε μία τέτοια διαδικασία.

Παρακάτω δίδεται ο αλγόριθμος του $Dijkstra$.

Algorithm 8 DijkstraAlgorithm

Data: G=(V,E,w), source
Result: dist[], prev[]
/* initialization */
dist[source] = 0 /* distance from source to source */
prev[source] = undefined /* previous node in optimal path initialization */
$Q = \emptyset$
for $v \epsilon V$ **do**
  if $v \neq source$ **then**
    $dist[v] = \infty$
    $prev[v] = undefined$
  end
  $Q = Q \cup \{v\}$
end
/* main process */
while $Q \neq \emptyset$ **do**
  $u = vertex\ in\ Q\ with\ minimum\ dist[u]$
  $Q = Q - \{u\}$
  for $v : (u, v) \epsilon E$ **do**
    $alt = dist[u] + w(u, v)$
    if $alt < dist[v]$ **then**
      $dist[v] = alt$
      $prev[v] = u$
    end
  end
end
return *dist[], prev[]*

Η αφαίρα ακτίνας r γύρω από την κορυφή v, συμβολιζεται ως $B(r, v)$, και είναι το σύνολο των κορυφών που η απόστασή τους είναι το πολύ r από την κορυφή v. Το κέλυφος της μπάλας $(ball\ shell)$ ακτίνας r γύρω από την κορυφή v, συμβολιζεται ως $BS(r, v)$, και είναι το σύνολο των κορυφών ακριβώς έξω από το $B(r, v)$. Δηλαδή, το σύνολο $BS(r, v)$ αποτελείται από οποιαδήποτε κορυφή $u \in (V - B(r, v))$ η οποία έχει μία γειτονική κορυφή $y \in B(r, v)$ έτσι ώστε $dist(u, v) = dist(v, y) + w(u, y)$.

Τέλος, η διάσπαση άστρου $(star-decomposition)$ είναι μία πολυμερής διαμέριση $\{V_0, V_1, ..., V_k\}$ ενός βεβαρημένου, συνδεδεμένου γράφου $G = (V, E, w)$ με κέντρο $x_0 \imath V$ αν $x_0 \epsilon V_0$ και

1. για κάθε $0 \leq i \leq k$, ο υπογράφος που προκύπτει από το V_i να είναι συνδεδεμένος και

2. για κάθε $i \geq 1$, το σύνολο V_i περιέχει μία κορυφή άγκυρα ($anchor$) x_i η οποία συνδέεται σε μία κορυφή $y_i \epsilon V_0$ μέσω μίας ακμής $(x_i, y_i) \epsilon E$. Η ακμή (x_i, y_i) καλείται γέφυρα ($bridge$) μεταξύ των συνόλων V_0 και V_i.

Εάν $r = rad_G(x_0)$, και $r_i = rad_{V_i}(x_i)$ για κάθε $0 \leq i \leq k$, τότε για $\delta, \varepsilon \ \leq \frac{1}{2}$ η διάσπαση άστρου $\{V_0, V_1, ..., V_k\}$ είναι μια (δ, ε)-διάσπαση άστρου αν

1. $r_0 \leq (1 - \delta)r$,

2. $dist(x_0, x_i) \geq \delta r$ για κάθε $i \geq 1$, και

3. $dist(x_0, x_i) + r_i \leq (1 + \varepsilon)r$ για κάθε $i \geq 1$.

Ας προχωρήσουμε τώρα στην ανάλυση των αλγορίθμων για την κατασκευή του επικαλύπτοντος δέντρου χαμηλής έκτασης. Ο πρώτος αλγόριθμος που θα δούμε είναι ο βασικός αλγόριθμος για την κατασκευή του δέντρου.

Algorithm 9 LowStretchTree

Data: G=(V,E,w), x_0
Result: T
if $|V| \leq 2$ **then**
  return G
end
$(V_0, ..., V_k, \vec{x}, \vec{y}) = StarDecomposition(G, x_0, \frac{1}{3}, \beta)$
for $i = 0, 1, ..., k$ **do**
  $T_i = LowStretchTree(G(V_i), x_i)$
end
$T = (\cup_i T_i) \cup (\cup_i (y_i, x_i))$
return T

Ο αλγόριθμος αυτός είναι αναδρομικός. Την πρώτη φορά καλείται για τον αρχικό γράφο $G = (V, E, w)$ και για μια κορυφή έναρξης την οποία θα προσδιορίζει ο χρήστης. Η τιμή της παραμετρού β είναι:

$$\beta = \frac{1}{2\log_{\frac{4}{3}} 2|V| + 32},$$

όπου το $|V|$ αφορά το πλήθος των κορυφών του αρχικού γράφου. Σε κάθε εκτέλεση λοιπόν του αλγορίθμου, ελέγχεται αρχικά η συνθήκη τερματισμού της αναδρομής: αν το πλήθος των κορυφών του γράφου εισόδου δεν είναι μεγαλύτερο του δύο τότε ο αλγόριθμος τερματίζει επιστρέφοντας τον γράφου εισόδου. Έπειτα, Καλείται ο αλγόριθμος $StarDecompo$ ο οποίος αναλαμβάνει να πραγματοποιήσει μια $(\frac{1/3}{\beta})$-διάσπαση άστρου επιστρέφοντας τα σύνολα κορυφών $V_0, V_1, ..., V_k$ καθώς και τα διανύσματα $\vec{x}$, $\vec{y}$ τα στοιχεία $\vec{x}(i), \vec{y}(i)$ των οποίων αντιστοιχούν στις ακμές-γέφυρες ($bridge\ edges$) που περιγράψαμε παραπάνω. Στην συνέχεια ο αλγόριθμος καλείται αναδρομικά για τους υπογράφους που προκύπτουν από τα σύνολα V_i. Τέλος ο αλγόριθμος σχηματίζει το επιθυμητό δέντρο ενώνοντας τα αποτελέσματα των αναδρομικών κλήσεων (τα οποία είναι ουσιαστικά υπόδεντρα του αποτελέσματος) με τις ακμές γέφυρες.

Ο επόμενος αλγόριθμος που θα δούμε είναι ο αλγόριθμος της $StarDecomposition$. Ως είσοδοι στον αλγόριθμο δίνονται ο γράφος $G = (V, E, w)$ στον οποίο επιθυμούμε να γίνει η διάσπαση, η κεντρική κορυφή x_0 και οι τιμές δ και ε που προσδιορίζουν την (δ, ε)-διάσπαση.

Algorithm 10 *StarDecomposition*

Data: G=(V,E), x_0, δ, ε
Result: $\{V_0, ..., V_k\}, \vec{x}, \vec{y}$
$\rho = rad_G(x_0)$
$r_0 = BallCut(G, x_0, \rho, \delta)$
$V_0 = B(r_0, x_0)$
$S = BS(r_0, x_0)$
$G' = (V', E', w') = G(V - V_0)$ /* the weighted graph induced by V-V_0 * /
$(\{V_1, ..., V_k\}, \vec{x}) = ConeDecomposition(G', S, \frac{\varepsilon\rho}{2})$
for *i = 1, 2, ..., k* **do**
| y_i = {v ϵV_0 | $(x_i, y_i)\epsilon E$ *and* y_i *is on the shortest path from* x_0 *to* x_i}
end
$\vec{y} = \{y_1, y_2, ..., y_k\}$
return *{*$V_0, ..., V_k\}, \vec{x}, \vec{y}$

Το πρώτο βήμα του αλγορίθμου είναι να υπολογίσει την ακτίνα του γράφου G σε σχέση με την κορυφή x_0. Έπειτα, ο αλγόριθμος $BallCut$ αναλαμβάνει να υπολογίσει την κατάλληλη ακτίνα r_0, έτσι ώστε στο επόμενο βήμα να υπολογστεί το πρώτο σύνολο V_0 που είναι η σφαίρα ακτίνας r_0 γύρω από την κορυφή x_0. Στα δύο επόμενα βήματα, το σύνολο S ορίζεται να είναι το κέλυφος της σφαίρας ακτίνας r_0 γύρω από την κορυφή x_0, και ο γράφος G' να είναι ο γράφος που προκύπτει από το σύνολο κορυφών $V - V_0$. Οι πληροφορίες αυτές δίνονται ως παράμετροι στην συνάρτηση $ConeDecomposition$ η οποία αναλαμβάνει να υπολογίσει τα υπόλοιπα σύνολα $V_1, V_2, ..., V_k$ καθώς και τις αντίστοιχες κορυφές $x_0, x_1, ..., x_k$. Στη συνέχεια με βάση τις κορυφές αυτές υπολοίζονται οι ακμές-γέφυρεςx_i, y_i. Οι κορυφές y_i είναι τέτοιες ώστε, η ακμή (x_i, y_i) να ανήκει στο σύνολο ακμών E του γράφου G και η εκάστοτε κορυφή y_i να βρίσκεται πάνω στο συντομότερο μονοπάτι από την κορυφή x_0 πος την κορυφή x_i.

Ο αλγόριθμος $BallCut$ είναι ένας αλγόριθμος αυξανόμενων συνόλων. Υλοποιεί μία καθιερομένη τεχνική την οποία εισήγαγε ο $Awerbuch$ για αυξανόμενες σφαίρες. Πρωτού αναλύσουμε τον αλγόριθμο, παραθέτουμε τον ορισμό ενός ομόκεντρου συστήματος ($concentric\ systems$).

Ένα ομόκεντρο σύστημα εντός ενός βεβαρημένου, συνδεδεμένου γράφου $G = (V, E, w)$ είναι μία οικογένεια συνόλων κορυφών $\Lambda = \{L_r \subseteq V : r\epsilon\Re_{\geq 0}$ για τα οποία ισχύει:

1. $L_0 \neq \emptyset$,
2. $L_r \subseteq L_{r'}$ για κάθε $r < r'$, και
3. αν μία κορυφή $u\epsilon L_r$ και $(u, v)\epsilon E$, τότε $v\epsilon L_{r+l(u,v)}$.

Για παράδειγμα, για κάθε κορυφή $x\epsilon V$, το σύνολο των σφαιρών $\{B(r, x)\}$ είναι ένα ομόκεντρο σύστημα. Έτσι λοιπόν προκύπτει ο παρακάτω αλγόριθμος.

Algorithm 11 *BallCut*

Data: G=(V,E,w), x, ρ, δ
Result: r
r = $\delta\rho$
while *cost(*$\partial(B(r, x))) > \frac{vol(B(r,x))+1}{(1-2\delta)\rho} \log_2(m + 1)$**do**
| *Find a vertex* $v \notin B(r, x)$ *that minimizes* $dist(x, v)$ *and set* $r = dist(x, v)$.
end
return *r*

Ο αλγόριθμος $StarDecomposition$ καλεί τον αλγόριθμο $ConeDecomposition$. Πρωτού αναλύσουμε τον αλγόριθμο $ConeDecomposition$, θα δώσουμε τον ορισμό των ιδανικών συνόλων $(ideals)$ και κώνων $(cones)$.

Για οποιοδήποτε βεβαρημένο γράφο $G = (V, E, w)$ και $S \subseteq V$ το σύνολο των εμπρόσθιων ακμών $(forward\ edge\ set)$ το οποίο προκαλεί το S ορίζεται ως:

$$F(S) = \{(u \to v) : (u, v) \epsilon E, dist(u, S) + l(u, v) = dist(v, S)\}$$

Για μία κορυφή $v \epsilon V$ το ιδανικό σύνολο της v $(ideal\ of\ v)$ που προκαλεί το S και συμβολίζεται ως $I_S(v)$, είναι το σύνολο των προσβάσιμων (από την v) κορυφών, μέσω κατευθυνόμενων ακμών του $F(S)$, συμπεριλαμβανομένης και της v.

Για μία κορυφή $v \epsilon V$, ο κώνος $(cone)$ πλάτους l γύρω από την v που προκαλείται από το S και συμβολίζεται ως $C_s(l, v)$ είναι το σύνολο κορυφών που ανήκουν στο V και είναι προσβάσιμές από την v μέσω ενός μονοπατιού, όπου το άθροισμα των βαρών (μηκών) των ακμών e που δεν ανήκουν στο $F(S)$ είναι το πολύ l. Προφανώς, $C_S(0, v) = I_S(v)$ για οποιαδήποτε κορυφή $v \epsilon V$.

Ο αλγόριθμος $ConeDecomposition$ αναλαμβάνει να υπολογίσει τους κώνους, δηλαδή τα σύνολα $\{V_1, V_2, ..., V_k\}$, καθώς και τις κορυφές άκγυρες $anchor\ vertices$ $q_1, q_2, ..., q_k$. Ως είσοδοι στον αλγόριθμο δίνονται ο αρχικός γράφος $G = (V, E, w)$, το σύνολο S με βάση το οποίο θα γίνει η διαμέριση σε κώνους και η παράμετρος Δ που σχετίζεται με την ακτίνα των κώνων.

Algorithm 12 ConeDecomposition

Data: G=(V,E,w), S, Δ
Result: $\{V_1, V_2, ..., V_k\}, \vec{x}$
$G_0 = G$
$S_0 = S$
$= 0$
while $S_k \neq \emptyset$ **do**
  $k = k + 1$
  Let x_k be an arbitrary vertex in S_{k-1} and set $r_k = ConeCut(G_{k-1}, x_k, 0, \Delta, S_{k-1})$.
  $V_k = C_{S_{k-1}}(r_k, x_k)$
  $G_k = G(V - \cup_{i=1}^{k} V_k)$
  $S_k = S_{k-1} - V_k$
end
$\vec{x} = \{x_1, x_2, ..., x_k\}$
return $\{V_1, V_2, ..., V_k\}, \vec{x}$

Ο αλγόριθμος καλείται από τον αλγόριθμο διάσπασης άστρου. Καθώς ο αλγόριθμος $StarDecomposition$ υπολογίζει την σφαίρα με κέντρο μία δεδομένη κορυφή, περνά ως παραμέτρους στην $ConeDecomposition$, τον εναπομείναντα γράφο -δηλαδή τον γράφο εκτός της σφαίρας- (παράμετρος G) και το κέλυφος της σφαίρας (παράμετρος S), το οποίο αντιστοιχεί στο σύνολο κορυφών με βάση το οποίο θα υπολογιστούν οι κώνοι. Ο αλγόριθμος επιλέγει επαναληπτικά τις κορυφές του συνόλου S και υπολογίζει τον αντίστοιχο κώνο, υπολογίζοντας πρώτα την σωστή ακτίνα μέσω του αλγορίθμου $ConeCut$. Οι κορυφές αυτές θα χρησιμοποιηθούν στις ακμές γέφυρες και για το λόγο αυτό, αποθηκεύονται στο διάνυσμα $\vec{x}$.

Ο αλγόριθμος $ConeCut$ είναι όμοιος με τον αλγόριθμο $BallCut$. Αναλαμβάνει να υπολογίσει την κατάλληλη ακτίνα για τον υπολογισμό ενός κώνου. Ως είσοδοι του αλγορίθμου δίνονται ο γράφος G, και πληροφορίες για τον κώνο που πρέπει να σχηματιστεί, δηλαδή το σύνολο κορυφών

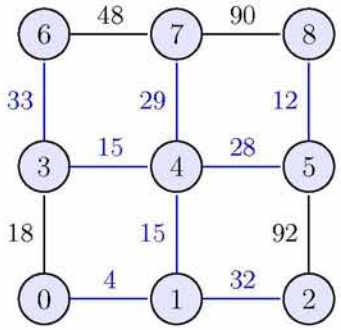
S και η κορυφή v και οι παράμετροι λ και λ' οι οποίες χρησιμοποιούνται ως αρχικές τιμές για τον υπολογισμό της ακτίνας.

Algorithm 13 *ConeCut*

Data:
Result:
$r = \lambda$
if *vol(E*$(C_S(\lambda, v))) == 0$**then**
  $\mu = (vol(C_S(r, v)) + 1)\log_2(m + 1)$
else
  $\mu = vol(C_S(r, v))\log_2(\frac{m}{vol(E(C_S(\lambda, v)))})$
end
while $cost(\partial(C_S(r, v))) > \frac{\mu}{\lambda' - \lambda}$ **do**
  $Find\ a\ vertex\ w \notin C_S(r, v)\ minimizing\ dist(w, C_S(r, v))\ and\ set\ r = r + dist(w, C_S(r, v))$
end

Για τον υπολογισμό του κώνου χρησιμοποιούμε τον ορισμό και υπολογίζουμε πρώτα το εμπρόσθιο σύνολο ακμών και κατόπιν το ιδανικό σύνολο της αντίστοιχης κορυφής. Επίσης, αποδεικνύεται οτι αν $G = (V, E, w)$ είναι ένας βεβαρημένος γράφος και $S \subseteq V$, τότε για κάθε κορυφή $v \epsilon V$, $\{C_S(l, v)\}_l$ είναι ένα ομόκεντρο σύστημα στον G.

Όπως και στην προηγούμενη παράγραφο ακολουθεί ένα σχήμα στο οποίο οι ακμές του υπαγράφου που ορίζει το επικαλύπτον δέντρο χαμηλής έκτασης χρωματίζονται με μπλε. Δίνεται επίσης και ο αντίστοιχος Λαπλασιανός πίνακας. Στο συγκεκριμένο παράδειγμα παρατηρούμε πως το επικαλύπτον δέντρο χαμηλής έκτασης έτυχε να ταυτίζεται με το ελάχιστο επικαλύπτον δέντρο και επομένως ταυτίζονται και οι πίνακες. Στην γενική πρίπτωση, αυτό δεν συμβαίνει. Σε ένα μεγαλύτερο παράδειγμα, μπορεί να φαίνεται χαρακτηριστικά η διαφορά.



$$L_H = \begin{bmatrix} 4 & -4 & & & & & & & \\ -4 & 51 & -32 & & -15 & & & & \\ & -32 & 32 & & & & & & \\ & & & 48 & -15 & & -33 & & \\ & -15 & & -15 & 87 & -28 & & -29 & \\ & & & & -28 & 40 & & & -12 \\ & & & -33 & & & 33 & & \\ & & & & -29 & & & 29 & \\ & & & & & -12 & & & 12 \end{bmatrix}$$

4.6 Επαυξημένο επικαλύπτον δέντρο χαμηλής έκτασης

Στην παράγραφο αυτή θα δούμε αλγορίθμους για την κατασκευή προρυθμιστών κατάστασης που προκύπτουν από επαυξημένα επικαλύπτοντα δέντρα. Θα αναλύσουμε τους αλγορίθμους των *Spielman* και *Teng* οι οποίοι προσθέτουν ακμές σε επικαλύπτοντα δέντρα χαμηλής έκτασης.

Ο αλγόριθμος που προσθέτει ακμές στο επικαλύπτον δέντρο χαμηλής έκτασης ονομάζεται $UltraSimple$ και είναι αρκετά αποδοτικός για επίπεδους γράφους. Οι επιπλέον ακμές επιλέγονται αφού πρώτα το δέντρο διασπαστεί σε έναν αριθμό από υπόδεντρα. Κατά την διάσπαση, τα υπόδεντρα αυτά μπορούν να έχουν μία κοινή κορυφή ή και να αποτελούνται από μία μοναδική κορυφή. Έτσι, για κάθε ζεύγος υπόδεντρων τα οποία συνδέονται μέσω ακμών του E, επιλέγεται μία από αυτές τις ακμές και προστίθεται στο δέντρο.

Σύμφωνα με τον ορισμό, δεδομένου ενός δέντρου T το οποίο επικαλύπτει ένα σύνολο κορυφών V, μία T-διάσπαση είναι μία διάσπαση του V σε σύνολα $W_1, W_2, ..., W_h$ έτσι ώστε $V = \cup_{i=1}^{h} W_i$. Ο γράφος που προκύπτει από το T σε κάθε W_i είναι ένα δέντρο, πιθανώς με μία μόνο κορυφή, και για όλα τα $i \neq j$ να ισχύει $|W_i \cap W_j| \leq 1$.

Επίσης, για ένα επιπρόσθετο σύνολο ακμών E επί του V, μία (T, E)-διάσπαση είναι ένα ζεύγος $(\{W_1, ..., W_h\}, \rho)$ όπου $\{W_1, ..., W_h\}$ είναι μία T-διάσπαση και ρ είναι μία αντιστοίχιση κάθε ακμής του E σε ένα σύνολο ή σε ένα ζεύγος συνόλων του $\{W_1, ..., W_h\}$ έτσι ώστε για κάθε ακμή $(u, v) \epsilon E$ να ισχύει:

1. αν $\rho(u, v) = \{W_i\}$ τότε $\{u, v\} \subseteq W_i$ και
2. αν $\rho(u, v) = \{W_i, W_j\}$ τότε, είτε $u \epsilon W_i$ και $v \epsilon W_j$, είτε $u \epsilon W_j$ και $v \epsilon W_i$.

Βέβαια, καθώς τα σύνολα W_i και W_j μπορούν να έχουν κοινό στοιχείο, είναι πιθανό ότι: $\rho(u, v) = \{W_i, W_j\}$ με $u \epsilon W_i$ και $v \epsilon W_i \cap W_j$.

Οι *Spielman* και *Teng* αποδεικνύουν οτι μία T-διάσπαση του E μπορεί να βρεθεί γρήγορα, με λίγα υπόδεντρα έτσι ώστε το άθροισμα των εκτάσεων *stretch* σε κάθε άλλο υπόδεντρο (που δεν αποτελείται από μία μόνο κορυφή) δεν είναι πολύ μεγάλο. Το θεώρημα τους ισχύει για οποιαδήποτε μη-αρνητική συνάρτηση επί των ακμών και όχι μόνο για την έκταση *stretch* και επομένως διατυπώνεται στην πιο γενική μορφή, η οποία δηλώνει οτι υπάρχει ένας αλγόριθμος $TreeDecomposition$, γραμμικού κόστους, που σε είσοδο ένα σύνολο ακμών E επί ενός συνόλου κορυφών V, ένα επικαλύπτον δέντο T επί του V, μία συνάρτηση $\eta : E \to \Re^+$ και έναν ακέραιο $1 < t < \sum_{e \epsilon E} \eta(e)$ έχει έξοδο μια (T, E)-διάσπαση $(\{W_1, ..., W_h\}, \rho)$ τέτοια ώστε:

1. $h \leq t$
2. για όλα τα W_i με $|W_i| > 1$,
$$\sum_{e \epsilon E : W_i \epsilon \rho(e)} \eta(e) \leq \frac{4}{t} \sum_{e \epsilon E} \eta(e).$$

Για τεχνικούς λόγους οι *Spielman* και *Teng* ορίζουν την συνάρτηση η να είναι:

$$\eta(e) = max\{st_T(e), 1\} \quad and \quad \eta(E) = \sum_{e \epsilon E} \eta(e).$$

Δεδομένης μίας (T, E)-διάσπασης $(\{W_1, ..., W_h\}, \rho)$ ορίζεται η αντιστοίχιση

$$\sigma : \{1, 2, ..., h\} \times \{1, 2, ..., h\} \to E \cup \{undefined\}$$

οπότε και τίθεται

$$\sigma(i,j) = \begin{cases} arg\ max_{e:\rho(e)=\{W_i,W_j\}} w(e)/\eta(e), \text{ αν η ακμή } i \neq j \text{ και υπάρχει τέτοια ακμή } e \\ undefined, \text{ αλλιώς} \end{cases}$$

Σε περίπτωση όπου δύο ακμές είναι ισόπαλες θεωρείται e η αλφαβητικώς μικρότερη ακμή που μεγιστοποιεί την ποσότητα $w(e)/\eta(e)$ ώστε $\rho(e) = \{W_i, W_j\}$. Σημειώνουμε οτι $\sigma i, j$ είναι μία βεβαρημένη ακμή.

Η αντιστοίχιση σ μας λέει ποια ακμή από το σύνολο ακμών E που βρίσκεται μεταξύ των υπόδεντρων W_i και W_j θα προστεθεί στο T. Για κάθε i, j τέτοια ώστε η $\sigma(i,j)$ να ορίζεται και για κάθε $e \epsilon E$ τέτοια ώστε $\rho(e) = \{W_j, W_i\}$ ισχύει οτι

$$\frac{w(e)}{\eta(e)} \leq \frac{w(\sigma(i,j))}{\eta(i,j)}.$$

Ο αλγόριθμος λοιπόν με τον οποίο θα κατασκευαστεί ο προρυθμιστής κατάστασης ενός απαυξημένου δέντρου είναι ο ακόλουθος.

Algorithm 14 UltraSimple

Data: E, t
Result: A
T = LowStretchTree(E)
F = AugmentTree(T,E,t)
A = T ∪ F
return *A*

Ο αλγόριθμος αυτός αρχικά κατασκευάζει ένα επικαλύπτον δέντρο χαμηλής έκτασης για τον γράφο $G = (V, E)$ και στην συνέχεια μέσω του αλγορίθμου $AugmentTree$ βρίσκει επιπλέον ακμές προκειμένου να επαυξήσει το δέντρο. Ο αλγόριθμος για την επαύξηση του δέντρου ακολουθεί παρακάτω.

Algorithm 15 AugmentTree

Data: T, E, t
Result: F
for $e \in E$ **do**
  compute $\text{st}_T(e)$
end
$(\{W_1, ..., W_h\}, \rho)$ = TreeDecomposition(T,E,η,t)
$F = \emptyset$
for *j = 1, ... h* **do**
  for *i = 1, ... j* **do**
    if $\sigma(i,j)\ is\ defined$ **then**
      $F = F \cup \sigma(i,j)$
    end
  end
end
return *F*

Παρακάτω ακολουθεί ο αλγόριθμος $TreeDecomposition$. Ο αλγόριθμος πραγματοποιεί μία διαπέραση κατά βάθος $(depth - first\ traversal)$ του δέντρου και δημιουργεί με άπληστο τρόπο τα

σύνολα W_j εφόσον προσπίπτουν σε ακμές των οποίων το άθροισμα των τιμών της συνάρτησης η δεν ξεπερνά το κατώφλι ϕ. Η διαπέραση πραγματοποιείται από τον αλγόριθμο Sub. Ο αλγόριθμος Sub καλείται πρώτη φόρα για την ρίζα $root$ του δέντρου και πρωτού επεξεργαστεί την κορυφή που έχει δοθεί ως εισόδος, καλεί αναδρομικά τον εαυτό του για τα παιδία της κορυφής αυτής.

Algorithm 16 *TreeDecomposition*

Data: T, E, η, t
Result: $\{W_1, W_2, ..., W_h\}, \rho$
/* h, ρ, ϕ and the W_i's are treated as global variables */
h = 0
for $e \in E$ **do**
  $\rho(e) = 0$
end
$\phi = 2 \sum_{e \in E} \frac{\eta(e)}{t}$
(F,w,U) = Sub(r)
if $U \neq \emptyset$ **then**
  h = h + 1
  $W_h = U$
  for $e \in F$ **do**
    $\rho(e) = \rho(e) \cup \{W_h\}$
  end
end
return $\{W_1, W_2, ..., W_h\}, \rho$

Ο αλγόριθμος Sub επιστρέφει ένα σύνολο κορυφών U, μαζί με ένα σύνολο ακμών F που προσπίπτουν σε κορυφές του U, καθώς και το w που είναι το άθροισμα των η επί των ακμών του συνόλου F. Οι κορυφές του συνόλου U είναι αυτές στο υπόδεντρο με ρίζα την κορυφή v οι οποίες δεν συμπεριλήφθηκαν σε κάποιο σύνολο W_j. Καθώς ο αλγόριθμος Sub δημιουργεί τα σύνολα κορυφών U_{sub}, οι ακμές οι οποίες προσπίπτουν στις κορυφές αυτές αποθηκεύονται στο F_{sub}, και των τιμών της συνάρτησης η επί των ακμών αυτών, αποθηκεύεται στο w_{sub}. Όταν ο αλγόριθμος Sub δημιουργεί ένα σύνολο κορυφών W_j, οι κορυφές αυτές αποτελούν ένα υπόδεντρο του T. Υπάρχουν τρεις τρόποι να δημιουργηθεί ένα σύνολο W_j. Ο πρώτος τρόπος είναι όταν κάποιο υποσύνολο των παιδιών του v επιστρέψει σύνολα F_i των οποίων οι τιμές των στοιχείων υπό την συνάρτηση η αθροίζονται σε περισσότερο από ϕ. Στην περίπτωση αυτή ο αλγόριθμος ενώνει τα αντίστοιχα σύνολα κορυφών, μαζί με την κορυφή v σε ένα σύνολο W_j. Ο δεύτερος τρόπος είναι όταν το άθροισμα των των τιμών της συνάρτησης η επί των ακμών που προσπίπτουν στην v και στις κορυφές των συνόλων U_i υπερβαίνει το κατώφλι phi, αλλά είναι μικρότερο από την τιμή 2ϕ. Στην περίπτωση αυτή η κορυφή v και τα σύνολα U_i ενώνονται και σχηματίζουν τι σύνολο W_j. Τέλος όταν το άθροισμα των τιμών της συνάρτησης η επί των ακμών που προσπίπτουν στην v και στις κορυφές των συνόλων U_i είναι μεγαλύτερο από 2ϕ, τότε η κορυφή v προστίθεται μόνη της ως ένα σύνολο ($singleton\ set$) και επίσης δημιουργείται ένα ακόμα σύνολο το οποίο περιέχει την ένωση της v και των U_i.

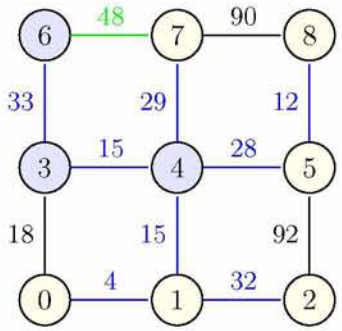
Algorithm 17 Sub

Data: v
Result: F, w, U
/* U is a set of vertices, F is the set of edges attached to U, and w is the sum of η over F */
Let v_1, v_2, ..., v_s *be the children of* v
$w_{sub} = 0, F_{sub} = \emptyset$, $U_{sub} = \emptyset$
for *i=1,2,...,s* **do**
  $(F_i, w_i, U_i) = Sub(v_i)$
  $w_{sub} = w_{sub} + w_i, F_{sub} = F_{sub} \cup F_i, U_{sub} = U_{sub} \cup U_i$
  if $w_{sub} \geq \phi$ **then**
    h = h + 1
    $W_h = U_{sub} \cup \{v\}$
    for $e \in F_{sub}$ **do**
      $\rho(e) = \rho(e) \cup \{W_h\}$
    end
    $w_{sub} = 0, F_{sub} = \emptyset$, $U_{sub} = \emptyset$
  end
end
$F_v = \{(u, v) \in E\}$ /* edges attached to v */
$w_v = \sum_{e \in F_v} \eta(e)$
if $\phi \leq w_v + w_{sub} \leq 2\phi$ **then**
  h = h + 1
  $W_h = U_{sub} \cup \{v\}$
  for $e \in F_{sub} \cup F_v$ **do**
    $\rho(e) = \rho(e) \cup \{W_h\}$
  end
  return $(\emptyset, 0, \emptyset)$
end
if $w_v + w_{sub} > 2\phi$ **then**
  h = h + 1
  $W_h = U_{sub} \cup \{v\}$
  for $e \in F_{sub}$ **do**
    $\rho(e) = \rho(e) \cup \{W_h\}$
  end
  h = h + 1
  $W_h = \{v\}$/ $*$ *create a singleton set* $*$ /
  for $e \in F_v$ **do**
    $\rho(e) = \rho(e) \cup \{W_h\}$
  end
  return $(\emptyset, 0, \emptyset)$
end
return $(F_{sub} \cup F_v, w_{sub} + w_v, U_{sub} \cup \{v\})$

Όπως είδαμε λοιπόν ο αλγόριθμος $TreeDecomposition$ απαιτεί και μία παράμετρο t. Σημειώνεται όμως οτι όταν $t \geq n$ τότε αλγόριθμος κατασκευάζει απλώς ένα σύνολο ($singleton\ set$) για κάθε κορυφή.

Στον ίδιο γράφο με τα προηγούμενα παραδείγματα, εάν εκτελέσουμε τον αλγόριθμο $UltraSimple$, πρώτα θα εξαχθεί το επικαλύπτον δέντρο χαμηλής έκτασης της προηγούμενης παραγράφου, μέσω του αλγορίθμου $LowStretchTree$, και στην συνέχεια καλείται ο αλγόριθμος $AugmentTree$ ο οποίος θα υπολογίσει τις επιπλέον ακμές που θα προστεθούν στο δέντρο. Μέσω του αλγορίθμου

TreeDecomposition ο γράφος θα χωριστεί σε δύο σύνολα κορυφών. Τα σύνολα αυτά είναι τα W_1={0,1,2,4,5,7,8} και W_2={3,4,6}. Επομένως, θα υπάρχει μία ακμή που θα ενώνει τα δύο αυτά σύνολα. Η ακμή αυτή επιλέγεται απο τον αλγόριθμο να είναι η $(6, 7)$. Στο επόμενο σχήμα σημειώνεται με μπλε χρώμα το επικαλύπτον δέντρο χαμηλής έκτασης και με πράσινο χρώμα η επιπρόσθετη ακμή.



Επίσης, ο αντίστοιχος Λαπλασιανός πίνακας L_H θα είναι ο ακόλουθος:

$$L_H = \begin{bmatrix} 4 & -4 & & & & & & & \\ -4 & 51 & -32 & & -15 & & & & \\ & -32 & 32 & & & & & & \\ & & & 48 & -15 & & -33 & & \\ & -15 & & -15 & 87 & -28 & & -29 & \\ & & & & -28 & 40 & & & -12 \\ & & & -33 & & & 81 & -48 & \\ & & & & -29 & & -48 & 77 & \\ & & & & & -12 & & & 12 \end{bmatrix}.$$

Κεφάλαιο 5

Αναδρομική επίλυση συστημάτων προρυθμιστών κατάστασης

Στο κεφάλαιο αυτό θα δούμε πως μπορούμε να επιλύσουμε με αναδρομικούς επιλυτές γραμμικά συστήματα. Όπως είδαμε μέχρι τώρα μπορούμε να λύσουμε αποτελεσματικά ένα γραμμικό σύστημα με την επαναληπτική μέθοδο συζηγών κλίσεων, χρησιμοποιώντας έναν αποδοτικό προρυθμιστή κατάστασης (*preconditioner*). Σε κάθε βήμα της μεθόδου επιλύεται ένα σύστημα του *preconditioner*. Ωστόσο ένα γραμμικό σύστημα του *preconditioner* μπορεί να ελλατωθεί σε ένα μικρότερο γραμμικό σύστημα, σε γραμμικό χρόνο, το οποίο μπορεί να επιλυθεί με αναδρομικό τρόπο. Στόχος του κεφαλαίου είναι να μελετήσουμε της μεθόδους και τους αντίστοιχους αλγορίθμους που μας επιτρέπουν την αναδρομική επίλυση γραμμικών συστημάτων προρυθμιστών κατάστασης.

5.1 Μερική παραγοντοποίηση *Cholesky*

Στόχος της μερικής παραγοντοποίησης *Cholesky* είναι να εξαλείψει γραμμές (ή ισοδύναμα στήλες, καθώς ο πίνακας είναι συμμετρικός) με ένα ή δύο μη-μηδενικά στοιχεία εκτός διαγωνίου. Οι γραμμές αυτές αντιπροσωπεύουν κόμβους που είναι ακροδέκτες ενός ή δύο κυκλωματικών στοιχείων.

Μέσω της διαδικασίας αυτής ο πίνακας B του *preconditioner* παραγοντοποιείται στην μορφή

$$B = PLCL^T P^T.$$

Ο πίνακας P είναι πίνακας μετάθεσης και χρησιμοποιείται για κατάλληλες εναλλαγές γραμμών και στηλών έτσι ώστε οι γραμμές με περισσότερα από δύο μη-μηδενικά στοιχεία εκτός διαγωνίου να είναι οι τελευταίες του πίνακα, φροντίζοντας ωστόσο ο πίνακας να παραμένει συμμετρικός. Ο πίνακας L είναι ο κάτω τριγωνικός και ο πίνακας C έχει την μορφή

$$\begin{bmatrix} D & 0 \\ 0 & A_1 \end{bmatrix}$$

Υποθέτοντας οτι ο αρχικός πίνακας B έχει διάσταση $dim\{B\} = n \times n$, και υπάρχουν n_1 γραμμές με περισσότερα από δύο μη μηδενικά στοιχεία εκτός διαγωνίου τότε ο πίνακας D έχει διάσταση $dim\{D\} = (n - n_1) \times (n - n_1)$ και αντίστοιχα ο πίνακας A_1 έχει διάσταση $dim\{A_1\} = n_1 \times n_1$. Ο αλγόριθμος που υλοποιεί την μερική παραγοντοποίηση *Cholesky* αναλαμβάνει να υπολογίσει τους

πίνακες P, L, D, A_1 και ορίζεται ως $(P, L, D, A_1) = PartialCholesky(B)$. Ο αλγόριθμος μπορεί να υλοποιηθεί έτσι ώστε να απαιτεί γραμμικό κόστος.

Ας υποθέσουμε οτι ο αρχικός πίνακας B μετά από τις μεταθέσεις γραμμών και στηλών μας δίνει τον πίνακα B_0 στον οποίο θα εφαρμόσουμε την παραγοντοποίηση. Κατά το πρώτο βήμα του αλγορίθμου έχουμε:

$$B = \begin{bmatrix} d_1 & \vec{v_1}^T \\ \vec{v_1} & B1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{\vec{v_1}}{d_1} & I_{n-1} \end{bmatrix} \begin{bmatrix} d_1 & 0 \\ 0 & B_1 - \frac{\vec{v_1}\vec{v_1}^T}{d_1} \end{bmatrix} \begin{bmatrix} 1 & \frac{\vec{v_1}^T}{d_1} \\ 0 & I_{n-1} \end{bmatrix} = L_1 C_1 L_1^T$$

Έπειτα ο πίνακας C_1 παραγοντοποιείται ξανά

$$C_1 = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & \vec{v_2}^T \\ 0 & \vec{v_2} & B_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{\vec{v_2}}{d_2} & I_{n-2} \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & B_2 - \frac{\vec{v_2}\vec{v_2}^T}{d_2} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{\vec{v_2}^T}{d_2} \\ 0 & 0 & I_{n-2} \end{bmatrix} = L_2 C_2 L_2^T$$

Η διαδικασία αυτή συνεχίζεται για όλες τις γραμμές με ένα ή δύο μη μηδενικά στοιχεία εκτός διαγωνίου και άρα απαιτεί συνολικά $n - n_1$ επαναλήψεις. Έτσι λοιπόν, ο πίνακας B_0 θα είναι:

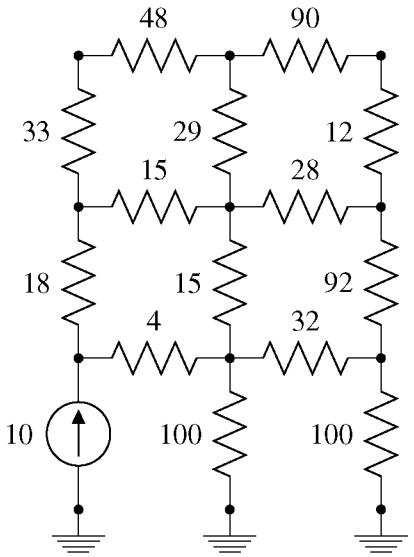
$$B_0 = L_1 L_2 ... L_{n-n_1} C L_{n-n_1}^T ... L_1^T L_2^T = L C L^T.$$

Συνυπολογίζοντας και τον πίνακα μετάθεσης P έχουμε την μερική παραγοντοποίηση $Cholesky$ του $preconditioner$ B θα έχουμε τελικά:

$$B = PLCL^T P^T.$$

Ας δούμε ένα ολοκληρωμένο παράδειγμα, πως από το γράφο του κυκλώματος εξάγουμε τον προρυθμιστή κατάστασης και πως αυτός παραγοντοποιείται μέσω της μερικής παραγοντοποίησης $Cholesky$.

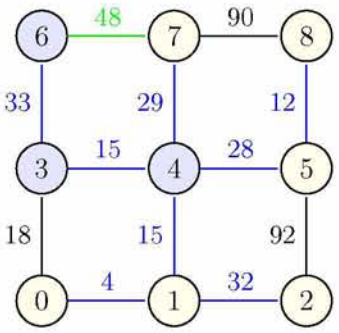
Αρχικά, ας υποθέσουμε το παρακάτω κύκλωμα:



Όπως είδαμε ο Λαπλασιανός πίνακας του κυκλώματος θα είναι ο

$$A = \begin{bmatrix} 22 & -4 & & -18 & & & & & \\ -4 & 151 & -32 & & -15 & & & & \\ & -32 & 224 & & & -92 & & & \\ -18 & & & 66 & -15 & & -33 & & \\ & -15 & & -15 & 87 & -28 & & -29 & \\ & & -92 & & -28 & 132 & & & -12 \\ & & & -33 & & & 81 & -48 & \\ & & & & -29 & & -48 & 167 & -90 \\ & & & & & -12 & & -90 & 102 \end{bmatrix},$$

ο οποίος είναι αντιστρέψιμος, λόγω των αντιστάσεων που συνδέονται με την γείωση. Δημιουργώντας τον *preconditioner* από το επαυξημένο επικαλύπτον δέντρο του γράφου του κυκλώματος,



χωρίς όμως να παραλείψουμε τις αντιστάσεις προς την γείωση, θα προκύψει ο αντιστρέψιμος πίνακας

$$B = \begin{bmatrix} 4 & -4 & & & & & & & \\ -4 & 151 & -32 & & -15 & & & & \\ & -32 & 132 & & & & & & \\ & & & 48 & -15 & & -33 & & \\ & -15 & & -15 & 87 & -28 & & -29 & \\ & & & & -28 & 40 & & & -12 \\ & & & -33 & & & 81 & -48 & \\ & & & & -29 & & -48 & 77 & \\ & & & & & -12 & & & 12 \end{bmatrix}.$$

Στην συνέχεια η μερική παραγοντοποίηση *Cholesky* θα μας δώσει τον πίνακα μετάθεσης P και τον κάτω τριγωνικό πίνακα L όπου:

$$P = \begin{bmatrix} 1 & & & & & & & & \\ & & & & & & & & 1 \\ & & & & & & & 1 & \\ & & & & & & 1 & & \\ & & & & & 1 & & & \\ & & & & 1 & & & & \\ & & & 1 & & & & & \\ & & 1 & & & & & & \\ & 1 & & & & & & & \end{bmatrix}, \quad L = \begin{bmatrix} 1 & & & & & & & & \\ & 1 & & & & & & & \\ & & 1 & & & & & & \\ & & & 1 & & & & & \\ & & & & 1 & & & & \\ & -1 & & & & 1 & & & \\ & & & -0.688 & -0.623 & & 1 & & \\ & & & -0.313 & -0.337 & -1 & -1 & 1 & \\ -1 & & -0.242 & & & & & & 1 \end{bmatrix},$$

έτσι ώστε ο *preconditioner* B να παραγοντοποιείται στο γινόμενο:

$$B = PL \begin{bmatrix} 4 & & & & & & & & \\ & 12 & & & & & & & \\ & & 132 & & & & & & \\ & & & 48 & & & & & \\ & & & & 77 & & & & \\ & & & & & 28 & & & \\ & & & & & & 28.39 & & \\ & & & & & & & 15 & -15 \\ & & & & & & & -15 & 139.242 \end{bmatrix} L^T P^T$$

Εύκολα δικρίνουμε τους πίνακες D και A_1 καθώς ο πίνακας D είναι:

$$D = \begin{bmatrix} 4 & & & & & & & \\ & 12 & & & & & & \\ & & 132 & & & & & \\ & & & 48 & & & & \\ & & & & 77 & & & \\ & & & & & 28 & & \\ & & & & & & 28.39 & \end{bmatrix},$$

και ο πίνακας A_1 είναι:

$$A_1 = \begin{bmatrix} 15 & -15 \\ -15 & 139.242 \end{bmatrix}.$$

Στην επόμενη παράγραφο θα εξετάσουμε πως μπορούμε από αυτήν την μορφή να επιλύσουμε το σύστημα του *preconditioner* με αποδοτικό τρόπο.

5.2 Αλγόριθμοι ενός επιπέδου

Όπως είδαμε για να επιλυθεί ένα σύστημα του πίνακα A μέσω PCG, μπορούμε να εξάγουμε πρώτα τον *preconditioner* B, ο οποίος προκύπτει από ένα επαυξημένο επικαλύπτον δέντρο χαμηλής έκτασης, και στην συνέχεια να κάνουμε μερική παραγοντοποίηση $Cholesky$ από όπου προκύπτουν οι πίνακες $P,\ L,\ D,\ A_1$.

Μπορούμε, μέσω της παραγοντοποίησης αυτής, να λύσουμε το σύστημα του *preconditoiner* χρησιμοποιώντας οπίσθια αντικατάσταση για το σύστημα του πίνακα L^T, έπειτα να λύσουμε τα συστήματα των πινάκων D και A_1 ξεχωριστά με άμεσες μεθόδους και τέλος να λύσουμε το σύστημα του πίνακα L με εμπρόσθια αντικατάσταση.

Για την επίλυση ένω και κάτω τριγωνικών συστημάτων μπορούμε να χρησιμοποιήσουμε τις μεθόδους που μας παρέχει η βιβλιοθήκη $Csparse$. Ο πίνακας D είναι διαγώνιος και επομένως το σύστημα $D\vec{x} = \vec{y}$ μπορεί να λυθεί πολύ απλά, αφού κάθε στοιχείο του διανύσματος αποτελέσματος είναι:

$$\vec{x}(i) = \frac{\vec{b}(i)}{D(i,i)}.$$

Επίσης, για την επίλυση του συστήματος του πίνακα A_1 μπορούμε να χρησιμοποιήσουμε την παραγοντοποίηση $Cholesky$ εφόσον ο πίνακας A_1 είναι και αυτός συμμετρικός και θετικά ορισμένος. Τις μεθόδους για την παραγοντοποίηση $Cholesky$ παρέχονται απίσης από την βιβλιοθήκη $Csparse$.

Η επίλυση του συστήματος με αυτόν τον τρόπο μας εισάγει ενός ιδέα ενός αναδρομικού επιλυτή, καθώς μπορούμε για την επίλυση συτήματος του πίνακα A_1 μπορούμε να χρησιμοποιήσουμε ξανά *preconditioned conjugate gradient* με κατάλληλο προρυθμιστή κατάστασης.

5.3 Ο αναδρομικός επιλυτής

Ο αναδρομικός αλγόριθμος για την επίλυση γραμμικών συστημάτων ενός πίνακα A, υπολογίζει ένα προρυθμιστή κατάστασης B, μέσω της διαδικασίας εξαγωγής ενός επαυξημένου επικαλύπτοντος δέντρου και στην συνέχεια χρησιμοποιώντας μερική παραγοντοποίηση $Cholesky$ φτάνει στην επίλυση ενός συστήματος μικρότερης τάξης (του πίνακα A_1) το οποίο επιχειρεί να λύσει αναδρομικά.

Όπως είδαμε στο προηγούμενο κεφάλαιο, λόγω του ισομορφισμού μεταξύ Λαπλασιανών πινάκων και βεβαρημένων γράφων μπορούμε να συσχετίσουμε τον πίνακα A_1 με ένα γράφο $G_1 = (V_1, E_1, w_1)$, να εξάγουμε ένα επαυξημένο επικαλύπτον δέντρο ως προρυθμιστή κατάστασης B_1 και να επιλύσουμε το σύστημα χρησιμοποιώντας PCG.

Έπειτα, αν κάνουμε μερική παραγοντοποίηση $Cholesky$ στον B_1 θα καταλήξουμε ξανά σε ένα σύστημα μικρότερης τάξης A_2. Η διαδικασία αυτή συνεχίζεται έως ότου καταλήξουμε σε έναν πίνακα A_k του οποίου η διάσταση είναι μικρότερη από από ένα κατώφλι ($threshold$). Ένας τέτοιος πίνακας είναι αρκετά μικρός ώστε να μπορούμε να τον επιλύσουμε με μία άμεση μέθοδο, που στην περίπτωσή μας θα είναι η παραγοντοποίηση $Cholesky$.

Φυσικά, όλοι οι απαραίτητοι προρυθμιστές κατάστασης και οι αντίστοιχες παραγοντοποιήσεις αυτών υπολογίζονται μία φορά πρωτού ξεκινήσει η επίλυση του γραμμικού συστήματος.

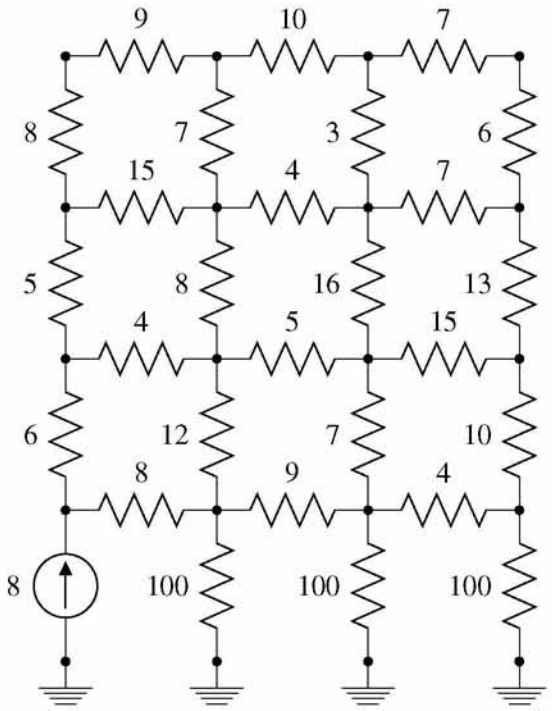
Algorithm 18 BuildPreconditioners

Data: A_0
while $dim\{A_i\} > threshold$ **do**
    i = i + 1
    $B_i = ASTPrecond(A_{i-1})$
    $(P_i, L_i, A_i) = PartialCholesky(B_i)$
end

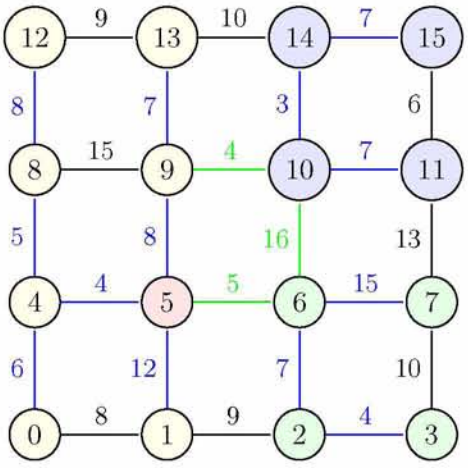
Ο αναδρομικός επιλυτής θα χρησιμοποιεί κάθε πίνακα B_i ως προρυθμιστή κατάστασης του πίνακα A_{i-1}. Άντί όμως να λυθεί το σύστημα B_i με άμεση μέθοδο, μέσω της μερικής παραγοντοποίησης $Cholesky$ θα μειωθεί σε ένα σύστημα μικρότερης τάξης A_i το οποίο θα επιλυθεί αναδρομικά. Η μέθοδος επίλυσης είναι η *preconditioned conjugate gradient*.

Ας δούμε όμως με ένα μεγαλύτερο παράδειγμα πως δουλεύει ο αναδρομικός επιλυτής. Θα θεωρήσουμε ένα κύκλωμα το οποίο απαρτίζεται από ένα πλέγμα εικοσιτεσσάρων αντιστάσεων και τριών επιπλέον αντιστάσεων προς την γείωση. Επίσης, θα θεωρήσουμε και μία πηγή ρεύματος ως διέγερση του κυκλώματός μας.

Από το κύκλωμα αυτό εξάγουμε τον πίνακα A_0 και τον *preconditioner* αυτού, έστω B_0, ο οποίος προκύπτει από το επαξημένο δέντρο χαμηλής έκτασης του γράφου του κυκλώματος. Αφού εξαχθεί το επικαλύπτον δέντρο χαμηλής έκτασης, ο γράφος χωρίζεται σε τέσσερα σύνολα κορυφών.



Τα σύνολα αυτά είναι τα W_1={0,1,4,5,8,9,12,13}, W_2={2,3,6,7}, W_3={10,11,14,15} και W_4={5}. Το τέταρτο σύνολο είναι ένα *singleton*, καθώς περιέχει μία μόνο κορυφή. Οι επιπλέον ακμές που επιλέγονται από τον αλγόριθμο να ενώσουν αυτά τα σύνολα είναι οι $(5,6)$ $(6,10)$ και $(9,10)$ τις οποίες χρωματίζουμε με πράσινο χρώμα κατ' αντιστοιχίαν με τα προηγούμενα σχήματα.

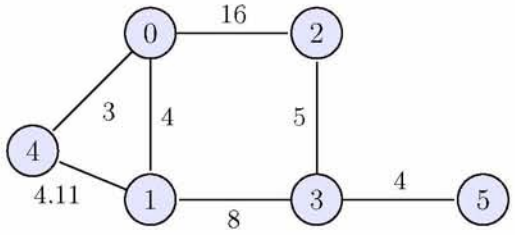


Να σημειώσουμε επίσης οτι η ακμή $(5,6)$ ανήκει στο επικαλύπτον δέντρο δέντρο χαμηλής έκτασης, παρ'όλα αυτά επιλέγεται καθώς ικανοποιεί το κριτήριο για την προσθήκη ακμής μεταξύ δύο συνόλων W_i και W_j. Μετά την διαδικασία εξαγωγής κατασκευάζεται ο πίνακας του B *preconditioner* και παραγοντοποιείται με μερική παραγοντοποίηση *Cholesky* στην μορφή $B_0 = P_0 L_0 C_0 L_0^T P_0^T$. Ό-

που τα *blocks* του πίνακα C_0 είναι τα D_0 και A_1. Πιο συγκεκριμένα ο A_1 είναι:

$$A_1 = \begin{bmatrix} 23 & -4 & -16 & & -3 \\ -4 & 16.11 & & -8 & -4.11 \\ -16 & & 25.55 & -5 & \\ & -8 & -5 & 27.7 & -4 \\ -3 & -4.11 & & & 7.11 \\ & & & -4 & & 4 \end{bmatrix}$$

Ας υποθέσουμε οτι η διάσταση του A_1 είναι αρκετά μεγάλη ώστε να μας επιτρέψει να συνεχίσουμε την αναδρομική διαδικασία. Επομένως, για να επιλύσουμε το σύστημα του A_1 θα χρησιμοποιήσουμε την (*preconditioned conjugate gradient*). Εύκολα παρατηρούμε οτι ο A_1 είναι Λαπλασιανός πίνακας και επομένως λόγω του ισομορφισμού με βεβαρημένους, μη-κατευθυνόμενους γράφους μπορούμε να τον αντιστοιχίσουμε στον παρακάτω γράφο:



Θα εξάγουμε τον *preconditioner* του A_1, έστω B_1, υπολογίζοντας ένα επαυξημένο δέντρο χαμηλής έκτασης του παραπάνω γράφου. Κατόπιν παραγοντοποιούμε τον τον B_1 με μερική παραγοντοποίηση *Cholesky* ώστε $B_1 = P_1 L_1 C_1 L_1^T P_1^T$, όπου

$$\begin{bmatrix} D_1 & 0 \\ 0 & A_2 \end{bmatrix}.$$

Υποθέτωντας τώρα ότι $dim\{A_2\} \leq threshold$ ο A_2 παραγοντοποιείται με παραγοντοποίση *Cholesky*. Η διαδικασία αυτή είναι το αναδρομικό *preconditioning* το οποίο περιγράψαμε παραπάνω.

Με τον αντίστοιχο αναδρομικό τρόπο γινεται και η επίλυση του γραμμικού συστήματος. Στην πρώτη επανάληψη της *PCG* για την επίλυση συστήματις του A_0 με *preconditioner* τον B_0 θα χρειαστεί να λυθεί ένα σύστημα του B_0. Επιλύοντας διαδοχικά για τους παράγντες του B_0 όπως περιγράψαμε στην προηγούμενη παράγραφο, θα φτάσουμε στην επίλυση συστήματος του A_1. Τότε ξανακαλούμε την *PCG* για τον A_1 με *preconditioner* τον B_1. Τώρα, κατά την πρώτη επανάληψη της *PCG* θα χρειαστεί να επιλυθεί ένα σύστημα του B_1, ο οποίος έχει παραγοντοποιηθεί με *partial Cholesky factoriazation*. Επιλύοντας, διαδοχικά για τους παράγοντές του φτάνουμε στην επίλυση συστήματος του πίνακα A_2, το οποίο όμως θα λυθεί με την άμεση μέθοδο της παραγοντοποίησης *Cholesky*. Μπορούμε να φανταστούμε την συνέχεια της αναδρομικής επίλυσης.

Στο παράδειγμα αυτό το βάθος της αναδρομής είναι 1. Σε μεγαλύτερα συστήματα όπου ο αρχικός πίνακας είναι π.χ. διάστασης $dim\{A_0\} = 900 \times 900$, η αναδρομή φτάνει σε βάθος 4, και αντίστοιχα σε μεγαλύτερο βάθος όσο αυξάνεται το μέγεθος του πίνακα.

Κεφάλαιο 6

Πειραματική διαδικασία και αποτελέσματα

Στόχος του κεφαλαίου αυτού είναι να παρουσιάσουμε τα αποτελέσματα της πειραματικής διαδικασίας προκειμένου να εξάγουμε χρήσιμα συμπεράσματα σχετικά με την αποδοτικότητα των αλγορίθμων που περιγράψαμε στα κεφάλαια 4 και 5.

6.1 Περιγραφή της πειραματικής διαδικασίας

Όπως προαναφέρθηκε, τα αποτελέσματα αφορούν τον χρόνο εξαγωγής και παραγοντοποίησης του εκάστοτε *preconditioner*, τα συνολικά βήματα που χρειάστηκε η *conjugate gradient* για να συγκλίνει καθώς και τον αντίστοιχο χρόνο.

Καθώς θέλουμε να δείξουμε την κλιμάκωση των αλγορίθμων σε σχέση με την διάσταση του κυκλώματος, θα δημιουργήσουμε μια σειρά από κυκλώματα αυξανόμενου μεγέθους. Η γενική μορφή των κυκλωμάτων αυτών θα είναι αυτή που ήδη είδαμε στα παραδείγματα, δηλαδή, ένα πλέγμα αντιστάσεων (*resistors grid*), με μία πηγή ρεύματος και κάποιες αντιστάσεις προς την γείωση.

Ο κώδικας παίρνει από την γραμμή εντολών έναν ακέραιο N και δημιουργεί ένα αρχείο περιγραφής κυκλώματος (*netlist*) που περιγράφει ένα πλέγμα $N \times N$ κόμβων, ώστε να προκύψει ένας $N^2 \times N^2$ πίνακας. Για παράδειγμα, αν εκτελέσουμε:

$$./netlist\_generator.out \;\; 3$$

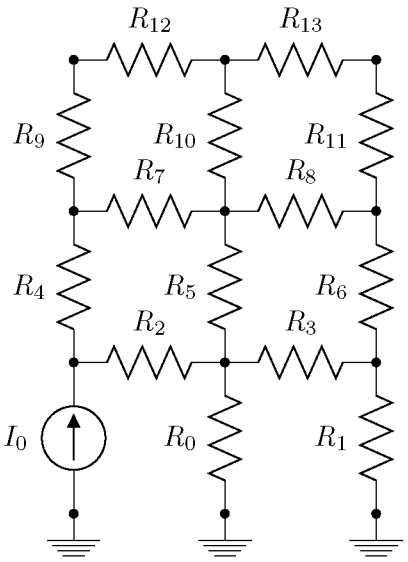
θα προκύψει το αρχείο $3x3grid.spice$ το οποίο θα περιγράφει το κύκλωμα της επόμενης σελίδας.

6.2 Πειραματική διαδικασία και ερμηνεία των αποτελεσμάτων

6.2.1 Χαρακτηριστικά του υπολογιστικού συστήματος

Πρωτού προχωρήσουμε στην πειραματική διαδικασία και στα αποτελέσματα αυτής, ας δούμε τα χαρακτηριστικά του υπολογιστικού συστήματος στο οποίο έγιναν οι μετρήσεις.

Όσο αναφορά το υλικό, ο υπολογιστής διαθέτει έναν επεξεργαστή Intel Core i7-3770 Processor @ 3.40GHz. Η αριτεκτονική του επεξεργαστή είναι x86 στα 64 bits και το byte order είναι little endian. Ο επεξεργαστής διαθέτει 4 υπολογιστικά cores ανά socket και υποστηρίζει 2 threads ανά core. Επίσης, τα χαρακτηριστικά των κρυφών μνημών cache του κάθε υπολογιστικού core είναι 32KB L1 data cache, 32KB L1 instruction cache, 256KB L2 cache και 8192KB L3 cache. Η συνολική μνήμη RAM που



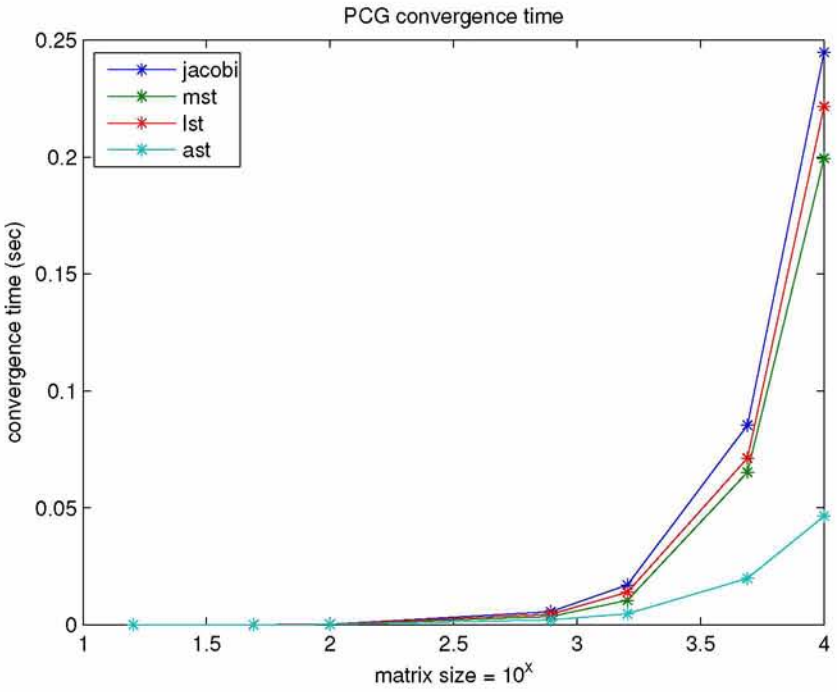
διατίθεται είναι 32 GB, από 4 φυσικές μνήμες Hynix χωρητικότητας 8GB, τύπου DDR3 σε συχνότητα 1600MHz.

Το λειτουργικό σύστημα είναι GNU/Linux διανομής Fedora έκδοσης 20 (Heisenbug). Η έκδοση του πυρήνα του λειτουργικού είναι 3.16.2-201.fc20.x86_64, για την αρχιτεκτονική x86 στα 64 bits.

Επίσης, καθώς ο κώδικας είναι γραμμένος σε C/C++ για την μεταγλώττιση του χρησιμοποιείται η συλλογή μεταγλωττιστών της GNU για C/C++, g++ (GNU C++ Compiler). Η έκδοση του gcc είναι gcc version 4.8.3 20140911 (Red Hat 4.8.3-7) (GCC). Να σημειώσουμε οτι κατά την μεταγλώττιση του κώδικα, δεν χρησιμοποιούμε καμία από τις βελτιστοποίησεις που μας παρέχονται από τον μεταγλωττιστή.

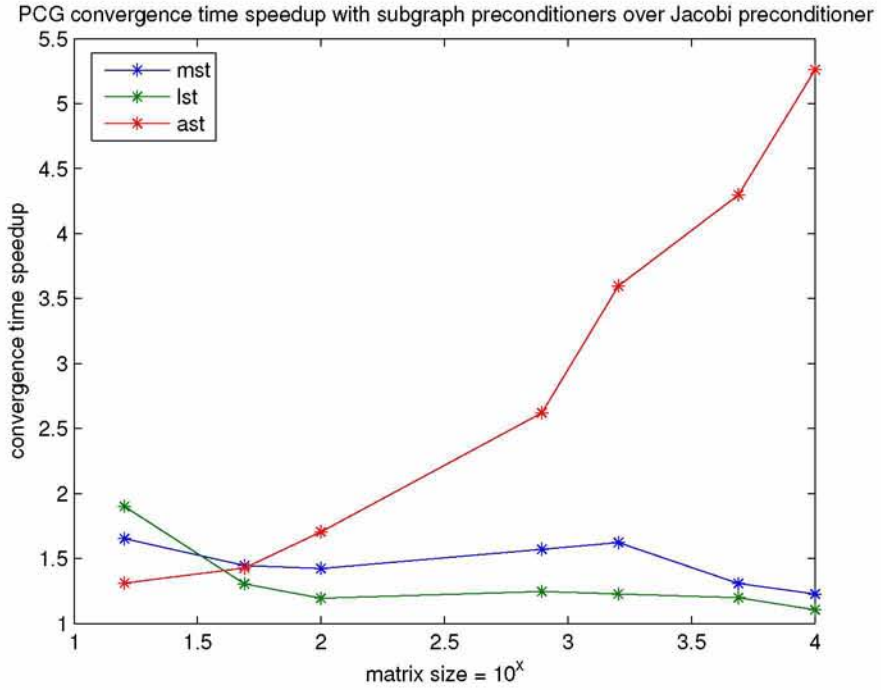
6.2.2 Πειραματική αξιολόγηση των προρυθμιστών κατάστασης

Η πρώτη γραφική παράσταση, μας δείχνει την απόδοση των προρυθμιστών κατάστασης καθώς κλιμακώνεται ο αριθμός των κόμβων του κυκλώματος και κατ΄ επέκταση, το μέγεθος του πίνακα.



Είναι εμφανές οτι ο προρυθμιστής κατάστασης επαυξημένου επικαλύπτοντος δέντρου χαμηλής έκτασης δίνει τον καλύτερο χρόνο στην σύγκλιση της μεθόδου σε κλιμάκωση του προβλήματος.

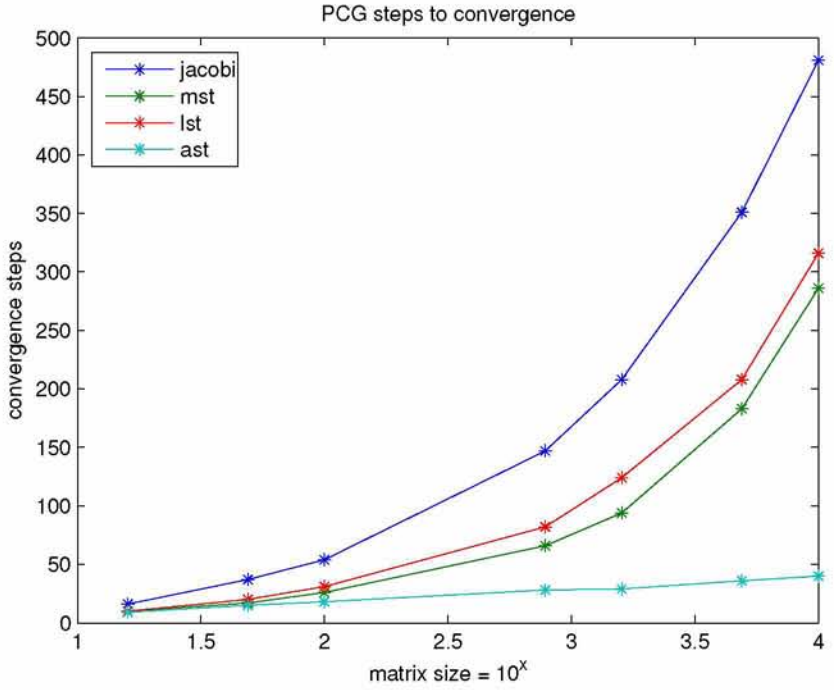
Αυτό φαίνεται και από το *speedup* του χρόνου εκτέλεσης. Στην επόμενη γραφική παράσταση μπορούμε να δούμε την χρονοβελτίωση που επιτυγχάνουν οι προρυθμιστές κατάστασης από την θεωρία γράφων σε σχέση με τον χρόνο σύγκλισης του *jacobi* προρυθμιστή κατάστασης.



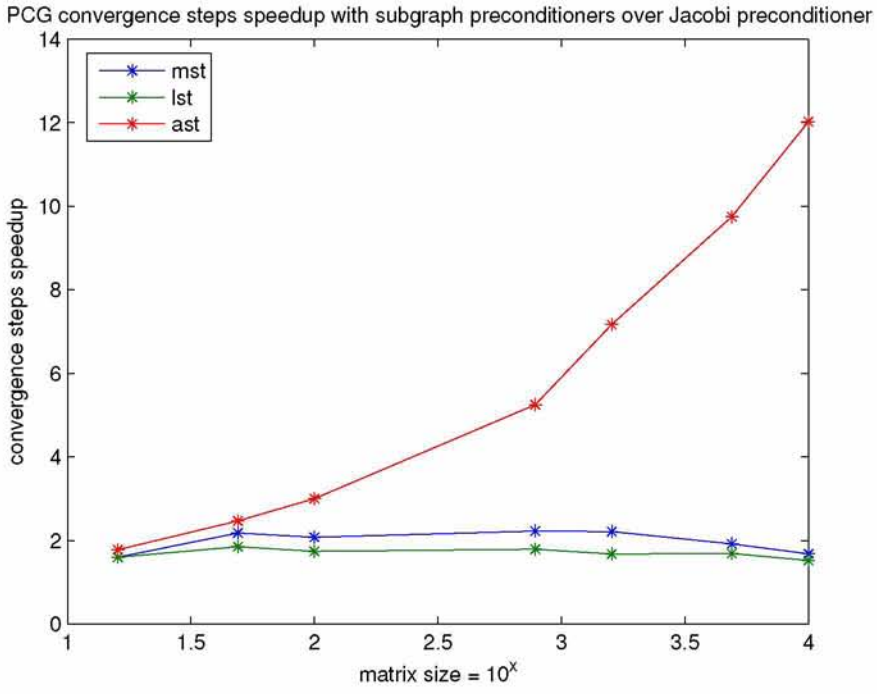
Καθώς κλιμακώνεται το μέγεθος του συστήματος, παρατηρούμε πως η χρονοβελτίωση που δίνουν οι προρυθμιστές κατάστασης ελάχιστου επικαλύπτοντος δέντρου και επικαλύπτοντος δέντρου χαμη-

λής έκτασης μειώνεται, ενώ αντιθέτως η χρονοβελτίωση του προρυθμιστή κατάστασης επαυξημένου επικαλύπτοντος δέντρου αυξάνεται.

Ας εξετάσουμε απίσης των αριθμό επαναλήψεων που χρειάστηκε η *conjugate gradient* για να υπολογίσει την λύση του συστήματος.

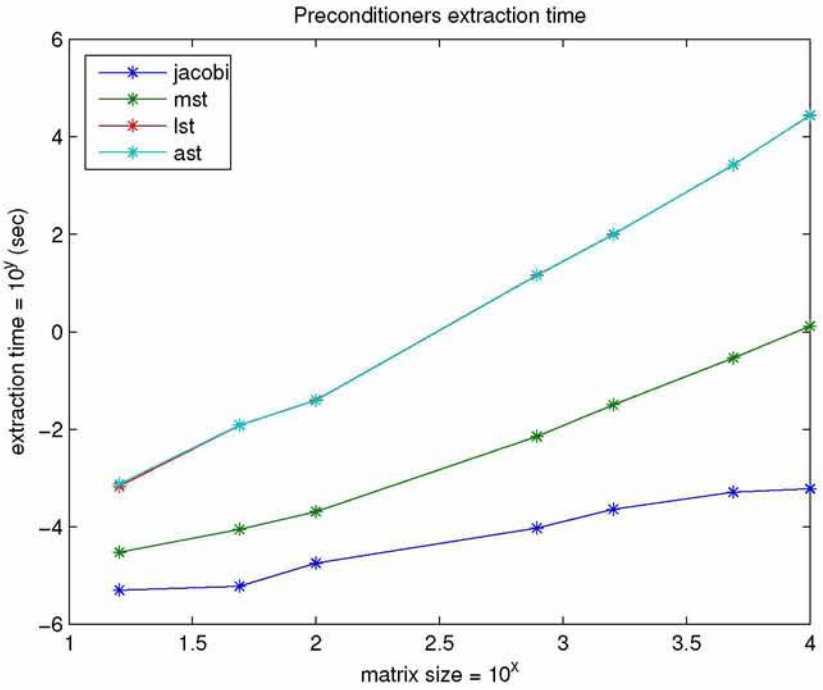


Ο προρυθμιστής κατάστασης έχει μειώσει σημαντικά τον αριθμό των επαναλήψεων και μπορούμε να δούμε και την γραφική παράσταση του *speedup* των επαναλήψεων της μεθόδου σε σχέση με τον με αυτές που απαιτεί ο *jacobi* προρυθμιστής κατάστασης.



Παρατηρούμε ξανά το αυξανόμενο *speedup* του προρυθμιστή κατάστασης επαυξημένου επικαλύπτοντος δέντρου. Σε επίλυση συστήματος μεγέθους $10^4 \times 10^4$ η μέθοδος γίνεται 12 φορές πιο γρήγορη ως προς το πλήθος των επαναλήψεων. Οι άλλοι δύο *preconditioners* σταθεροποιούνται σε *speedup* επαναλήψεων 2 καθώς κλιμακώνεται το πρόβλημα.

Τέλος ας δούμε το κόστος κατασκευής των προρυθμιστών κατάστασης.



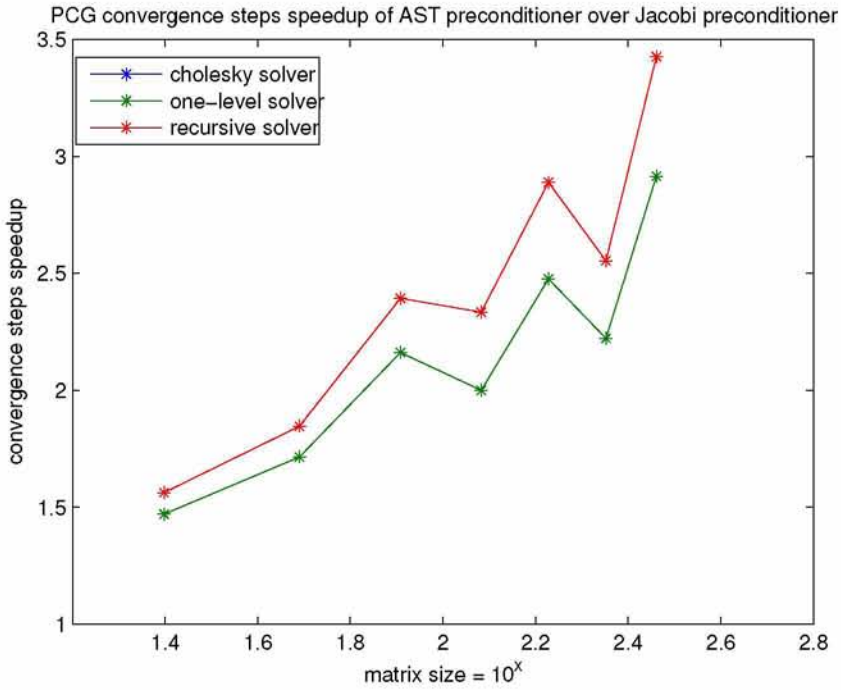
Η κλίμακα του χρόνου στο οποίο εξάγονται οι προρυθμιστές κατάστασης είναι λογαριθμική. Αντιλαμβανόμαστε λοιπόν πως το κόστος εξαγωγής του προρυθμιστή κατάστασης επικαλύπτντος δέντρου χαμηλής έκτασης είναι πολύ μεγάλο. Το κόστος κλιμακώνεται ανάλογα με το μέγεθος του προβλήματος. Έτσι λοιπόν, αν το μέγεθος του πίνακα αυξηθεί κατά μά τάξη μεγέθους, θα αυξηθεί κατά μία τάξη μεγέθους και το κόστος εξαγωγής του επικαλύπτοντος δέντρου χαμηλής έκτασης. Ο προρυθμιστής κατάστασης επαυξημένου επικαλύπτοντος δέντρου χαμηλής έκτασης έχει σχεδόν το ίδιο κόστος. Η διαδικασία της επαύξησης δεν είναι ιδιαίτερα χρονοβόρα, όπως αυτό φαίνεται και από τις γραφικές παραστάσεις.

Εν τέλει, οι προρυθμιστές κατάστασης που προέκυψαν από την θεωρία γράφων είναι εξαιρετικά αποδοτικοί. Ιδιαίτερα ο προρυθμιστής κατάστασης επαυξημένου επικαλύπτοντος δέντρου χαμηλής έκτασης, παρά το ακριβό κόστος εξαγωγής του, δίνει αυξανόμενο *speedup* στον απαιτούμενο αριθμό επαναλήψεων, αλλά και στον χρόνο σύγκλισης της μεθόδου.

6.2.3 Πειραματική αξιολόγηση του αναδρομικού επιλυτή

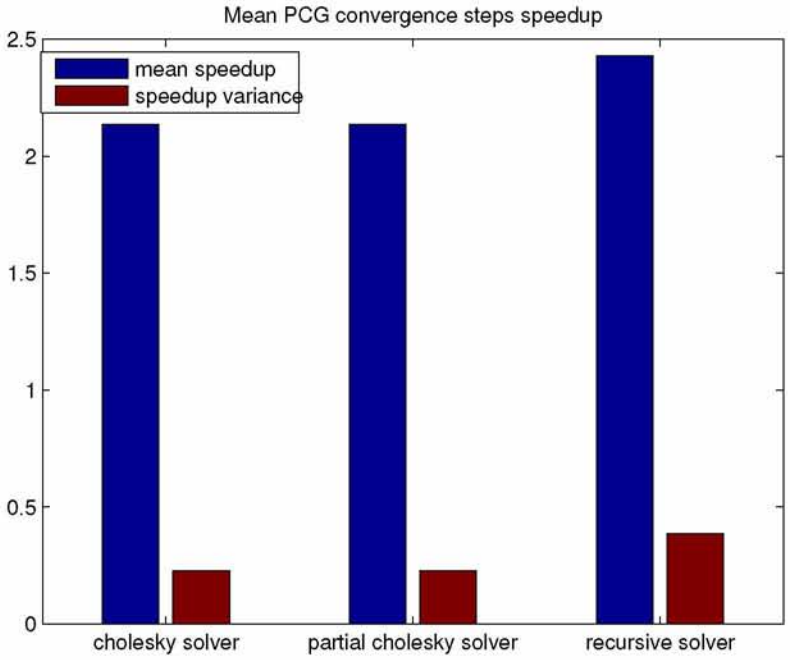
Στην υποπαράγραφο αυτή θα εξετάσουμε τα αποτελέσματα του επιλυτή ενός επιπέδου και του αναδρομικού επιλυτή. Καθώς και οι δύο τρόποι επίλυσης αφορούν το σύστημα του προρυθμιστή κατάστασης επαυξημένου επικαλύπτοντος δέντρου χαμηλής έκτασης, θα συγκρίνουμε την απόδοση των δύο επιλυτών σε σχέση με τον επίλυτη *Cholesky*.

Αρχικά ας δούμε τις επιπτώσεις στο *speedup* του απαιτούμενου αριθμού βημάτων σύγκλισης της μεθόδου συζυγών κλίσεων.

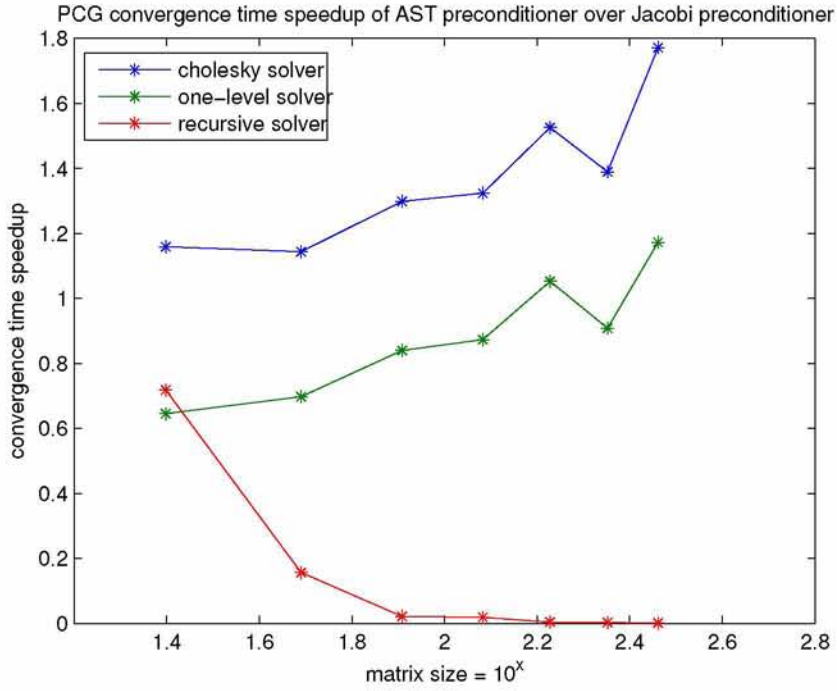


Ο επιλυτής ενός επιπέδου δεν προσφέρει ουσιαστική επιτάχυνση των βημάτων σύγκλισης, ενώ ο αναδρομικός επιλυτής προσφέρει ένα επιπλέον *speedup* σε σχέση με την επίλυση συστημάτων του *preconditioner* με παραγοντοποίηση *Cholesky*.

Καθώς οι διαστάσεις των πινάκων είναι κοντά ως προς την τάξη μεγέθους μπορούμε να δούμε και το μέσο *speedup* του αριθμού επαναλήψεων.

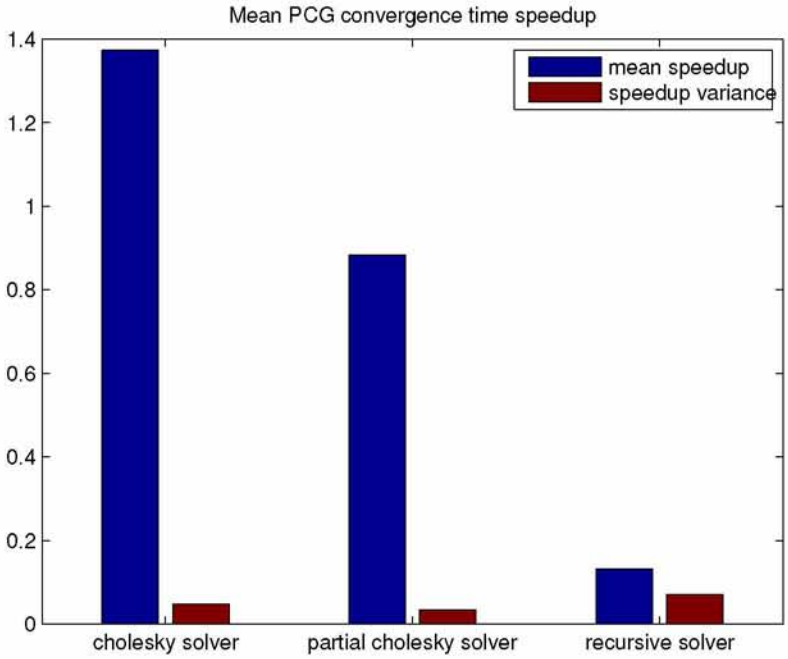


Επίσης ας δούμε το *speedup* στο χρόνο σύγκλισης της PCG.

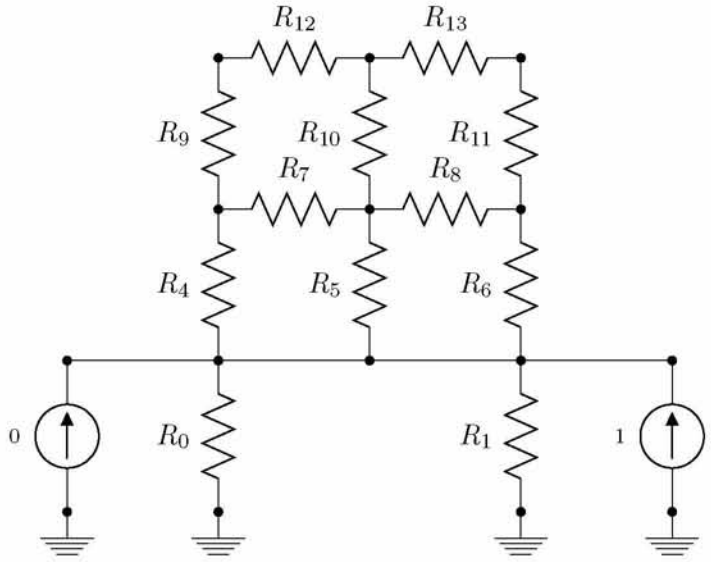


Παρατηρούμε πως και ο επιλυτής ενός επιπέδου και ο αναδρομικός επιλυτής δεν έχουν αποδοτικά αποτελέσματα στην περίπτωση αυτή. Το *speedup* λαμβάνει πολύ χαμηλές τιμές, καθώς ο χρόνος σύγκλισης της PCG, ιδιαίτερα με αναδρομική επίλυση του AST *preconditioner*, είναι πολύ μεγαλύτερος του χρόνου σύγκλισης της PCG με επίλυση του AST *preconditioner* με παραγοντοποίηση *Cholesky*.

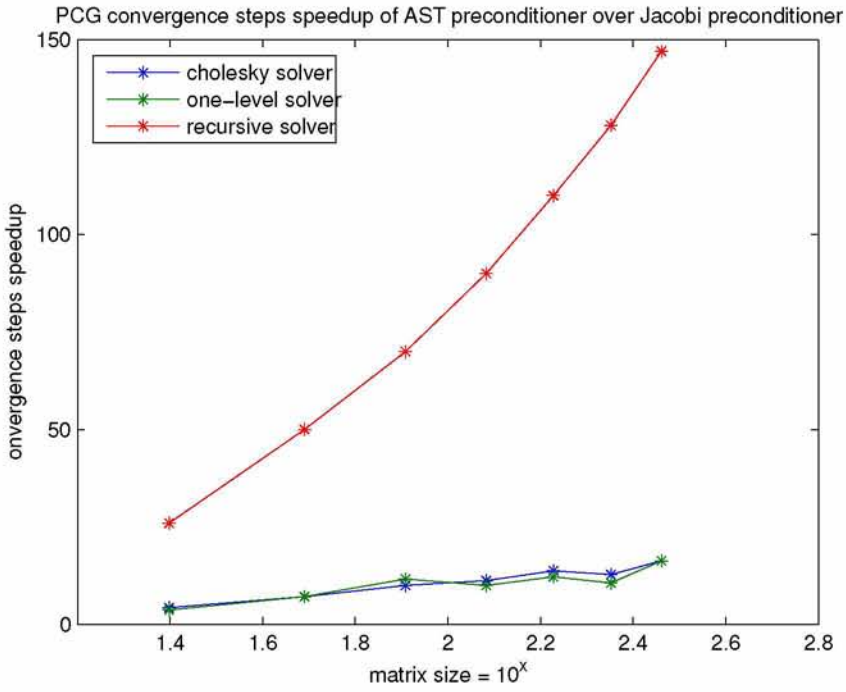
Ας επαληθεύσουμε όπως και πριν με το μέσο *speedup* του χρόνου σύγκλισης.



Ας εξετάσουμε και μία πολύ ενδιαφέρουσα περίπτωση. Υποθέτουμε την περίπτωση κυκλώματος όπως στην παράγραφο 6.1, ένα πλέγμα αντιστάσεων, χωρίς όμως επιπλέον αντιστάτες προς την γείωση. Θα θεωρήσουμε οτι στο κύκλωμα παρέχεται ενέργεια από πραγματικές πηγές ρεύματος, στις οποίες υπάρχει μια παράλληλη εσωτερική αντίσταση. Έτσι λοιπόν προκύπτει το ακόλουθο κύκλωμα. Παρατηρούμε πως αντιστάτες που είναι συνδεδεμένοι στην γείωση, αφορούν τους κόμβους στους οποίους παρέχεται ενέργεια από τις πηγές.

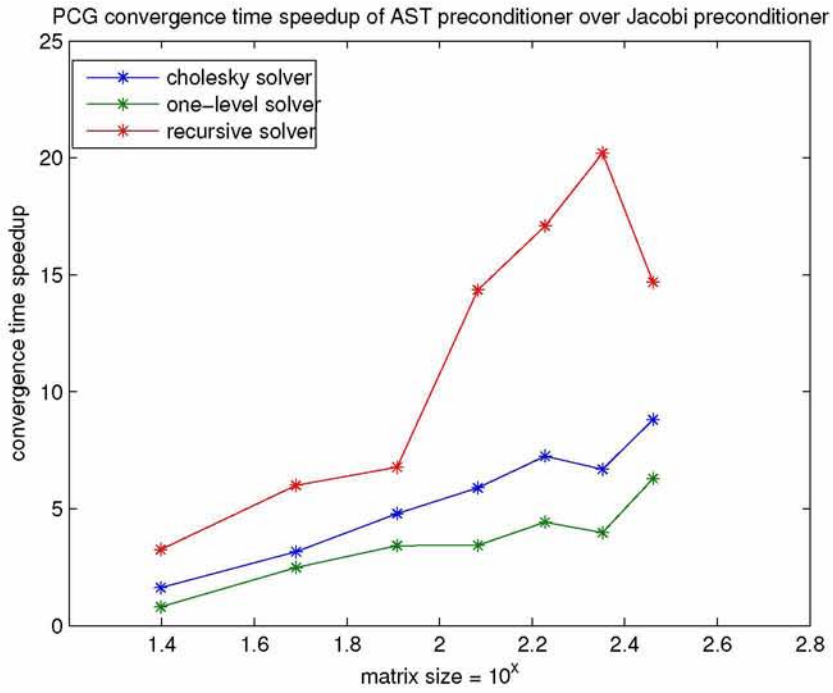


Σε αυτή την περίπτωση ο αναδρομικός επιλυτής είναι πολύ αποδοτικός. Ας δούμε αρχικά το *speedup* του απαιτούμενου αριθμού βημάτων σύγκλισης της μεθόδου συζυγών κλίσεων.



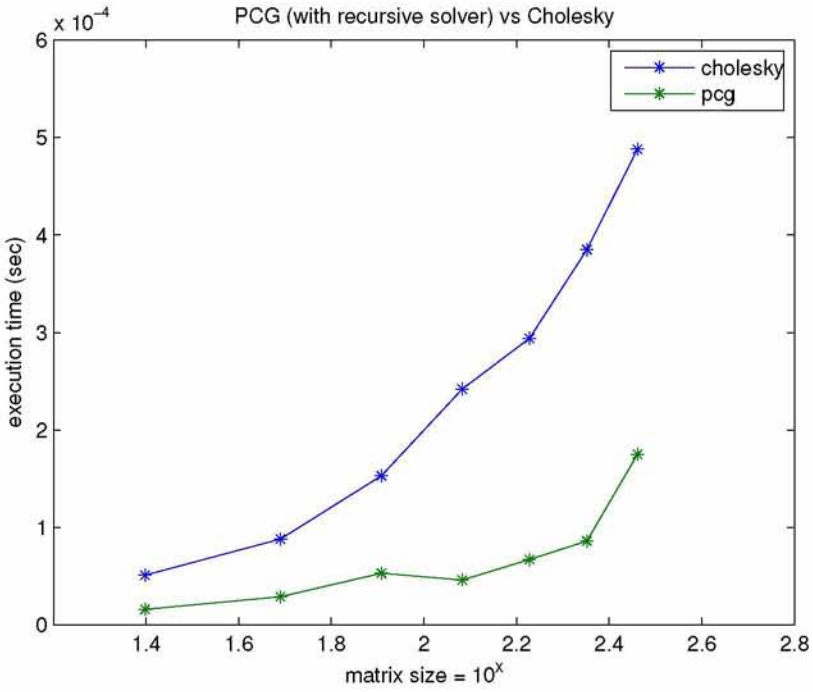
Στην περίπτωση αυτή ο αναδρομικός επιλυτής δίνει αυξανόμενο *speedup* στον αριθμό βημάτων σύγκλισης της *conjugate gradient*. Με την χρήση του αναδρομικού επιλυτή, σε αυτές τις τάξεις

μεγέθους, η *conjugate gradient* συγκλίνει σε μόλις 1 βήμα! Φυσικά, αυτό οδηγεί και σε τεράστιο *speedup* στον χρόνο σύγκλισης της μεθόδου.

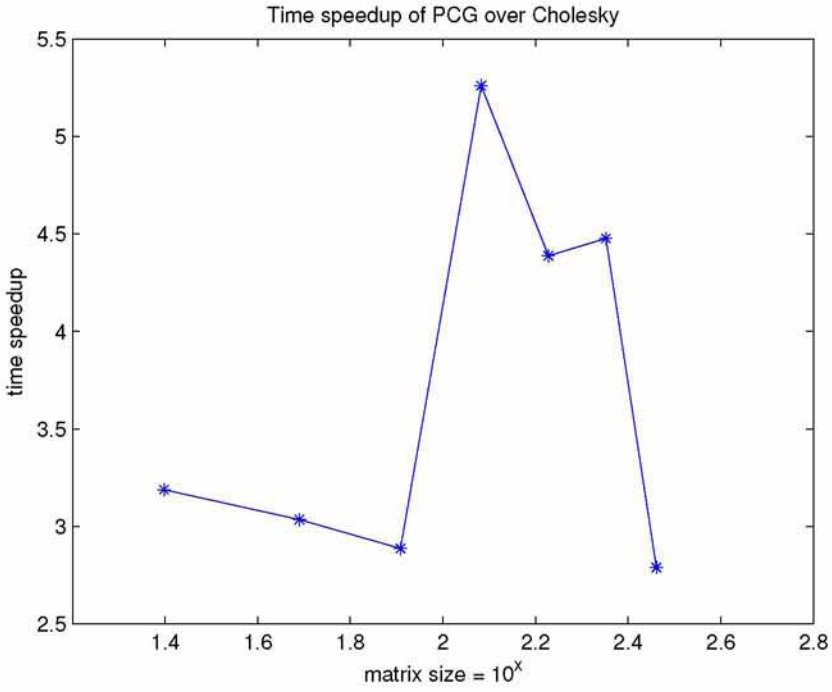


Με χρήση επαυξημένου επικαλύπτοντος δέντρου χαμηλής έκτασης και αναδρομικού επιλυτή για το σύστημα του *preconditioner* η *conjugate gradient* συγκλίνει τόσο γρήγορα που ο χρόνος επίλυσης του συστήματος είναι μικρότερος και από την χρήση άμεσης μεθόδου για την επίλυση του αρχικού συστήματος.

Ας δούμε στην παρακάτω γραφική παράσταση τους χρόνους επίλυσης του MNA συστήματος με παραγοντοποίηση $Cholesky$ και με PCG με *preconditioner* επαυξημένου επικαλύπτοντος δέντρου χαμηλής έκτασης και αναδρομικό επιλυτή.



Μπορούμε να δούμε και το *speedup* που επιτυγχάνεται στην ακόλουθη γραφική παράσταση.



6.3 Μελλοντικές επεκτάσεις

Όπως είδαμε στην προηγούμενη παράγραφο οι προρυθμιστές κατάστασης που περιγράψαμε αποδυκνύονται εξαιρετικά αποδοτικοί στην πράξη. Ωστόσο, η εξαγωγή ενός επικαλύπτοντος δέντρου χαμηλής έκτασης, και κατ' επέκταση ενός επαυξημένου επικαλύπτοντος δέντρου χαμηλής έκτασης μπορεί να αποβεί χρονοβόρα, καθώς απαιτεί μεγάλο υπολογιστικό κόστος.

Η υλοποίηση της κατασκευής ενός επικαλύπτοντος δέντρου χαμηλής έκτασης στηρίζεται στους αλγορίθμους των *Spielman* και *Teng* της εργασίας τους *'Lower Stretch SpanningTrees'*. Σε προγραμματιστικό επίπεδο όμως υπάρχουν πληροφορίες οι οποίες επαναϋπολογίζονται. Μπορούμε να αποφύγουμε τέτοιου είδους επανυπολογισμούς, αν αποθηκεύουμε πληροφορίες που θα χρειαστούμε αργότερα, εισάγοντας έτσι ένας είδος *caching*, σε αλγοριθμικό επίπεδο.

Επίσης, όπως προαναφέρθηκε, το πρόγραμμά μας εκτελείται χωρίς καθόλου βελτιστοποιήσεις. Μπορούμε επομένως να προβούμε σε μία σειρά βελτιστοποιήσεων του κώδικα, ώστε να επιταχυνθεί η ταχύτητα εκτέλεσης του. Μέσω $profiling$, μπορούμε να εντοπίσουμε τα *hot spots* ώστε να ξεκινήσουμε την διαδικασία της βελτιστοποίησης από τα συγκεκριμένα τμήματα κώδικα.

Επεκτείνοντας την ιδέα της βελτιστοποίησης, μπόρουμε να εφαρμόσουμε τεχνικές παραλληλοποίησης έτσι ώστε ο κώδικας να εκτελείται σε πολλαπλούς πυρήνες, επιτυγχάνοντας έτσι ακόμα ταχύτερη εκτέλεση του προγράμματός μας.

Σημειώσαμε, στην προηγούμενη παράγραφο, πως ο προρυθμιστής κατάστασης ένος ελάχιστου επικαλύπτοντος δέντρου του γράφου των αντιστάσεων, επιταχύνει περισσότερο την σύγκλιση της PCG από τον αντίστοιχο προρυθμιστή κατάσταση ένος επικαλύπτοντος δέντρου χαμηλής έκτασης του γράφου των αντιστάσεων. Είναι ενδιαφέρον να δημιουργήσουμε, μέσω ενός αποδοτικού αλγορίθμου, ένα επαυξημένο ελάχιστο επικαλύπτον δέντρο και κατόπιν να εφαρμόσουμε αναδρομική επίλυση του αντίστοιχου προρυθμιστή κατάστασης.

Τέλος, μπορούμε να επεκτείνουμε τους αλγορίθμους για την κατασκευή προρυθμιστών κατάστασης για γενικούς γράφους οι οποίοι δεν είναι επίπεδοι (*planar*). Στην εργασία αυτή οι γράφοι που προέκυψαν από επίπεδες κυκλωματικές διατάξεις, χωρίς όμως αυτή να είναι η γενικότερη περίπτωση. Στην γενικότερη περίπτωση μη επίπεδων γράφων πρέπει να εξάγουμε ένα $ultra - sparsifier$ υπογράφο, ο οποίος θα μας δώσει τον αντίστοιχο *preconditioner*. Επίσης, καθώς ένα κύκλωμα μπορεί να περιέχει πολύ μεγάλο αριθμό κόμβων μπορούμε να εφαρμόσουμε ιεραρχικό $suport - graph\ preconditioning$ ώστε να υπολογίσουμε τους υπογράφους συγκεκριμενών *blocks* του αρχικού γράφου, αντί για τον υπογράφο ολόκληρου του γράφου.

Βιβλιογραφία

1. Farid N. Najm. "Circuit simulation". Wiley, 2010.

2. Timothy A. Davis. "Direct methods for sparse linear systems". SIAM, 2006.

3. Richard Barrett, Michael Berry, Tony F. Chan, James Demmel, June M. Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk Van der Vorst. "Templates for the solution of linear systems". SIAM, 1993.

4. Pravin M. Vaidya. "Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners", unpublished manuscript UIUC 1990. A talk based on the manuscript was presented at the IMA Workshop on Graph Theory and Sparse Matrix Computation. October 1991, Minneapolis.

5. Daniel A. Spielman, and Shang-Hua Teng. "Nearly-Linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems". arXiv:cs/0607105v5 [cs.NA], 2012.

6. Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. "Lower-Stretch Spanning Trees". SIAM Journal on Computing, 38(2):608-628, 2008.

7. Noga Alon, Richard M. Karp, David Peleg, and Douglas West, "A graph-theoretic game and its application to the k-server problem". SIAM Journal on Computing, 24(1):78–100, 1995.

8. William N. Anderson and Thomas D. Morley. "Eigenvalues of the laplacian of a graph". Linear and Multilinear Algebra, 18(2):141–145, 1985.

9. Ittai Abraham and Ofer Neiman. "Using petal-decompositions to build a low stretch spanning tree". In Proceedings of the 44th Annual ACM Symposium on the Theory of Computing (STOC '12), pages 395–406, 2012.

10. O. Axelsson. "A survey of preconditioned iterative methods for linear systems of algebraic equations". BIT Numerical Mathematics, 25(1):165–187, March 1985.

11. Erik G. Boman and Bruce Hendrickson. "Support theory for preconditioning. SIAM Journal on Matrix Analysis and Applications", 25(3):694–717, 2003.

12. Doron Chen and Sivan Toledo. "Vaidya's preconditioners: implementation and experimental study". Electronic Transactions on Numerical Analysis, 16:30–49, 2003.

13. Keith Gremban. "Combinatorial preconditioners for sparse, symmetric, diagonally dominant linear systems". PhD thesis, Carnegie Mellon University, CMU-CS-96- 123, 1996.

14. I. Koutis, G.L. Miller, and R. Peng. Approaching optimality for solving sdd linear systems. In Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on, pages 235 –244, 2010.

15. Daniel A. Spielman and Shang-Hua Teng."Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems". In Proceedings of the thirty-sixth annual ACM Symposium on Theory of Computing, pages 81–90, 2004.

16. Daniel A. Spielman and Shang-Hua Teng. "Spectral sparsification of graphs". SIAM Journal on Computing, 40(4):981–1025, 2011.

17. Gilbert Strang. "Introduction to applied mathematics". Wellesley-Cambridge Press, 1986.

18. Daniel A. Spielman and Jaeoh Woo. "A note on preconditioning by low-stretch spanning trees". CoRR, abs/0903.2816, 2009.