ΕΘΝΙΚΌ ΜΕΤΣΌΒΙΟ ΠΟΛΥΤΕΧΝΕΊΟ
ΣΧΟΛΉ ΗΛΕΚΤΡΟΛΌΓΩΝ ΜΗΧΑΝΙΚΏΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΗΜΑΤΩΝ, ΕΛΕΓΧΟΥ ΚΑΙ ΡΟΜΠΟΤΙΚΗΣ

Ανάπτυξη Προγράμματος Ανάλυσης Συχνοτήτων και Υπολογισμού Ευαισθησιών σε Περιβάλλον MATLAB

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Ι. Δρυς

Επιβλέπων : Νικόλαος Γ. Μαράτος

Καθηγητής ΕΜΠ

Αθήνα, Οκτώβριος 2015

ΕΘΝΙΚΌ ΜΕΤΣΌΒΙΟ ΠΟΛΥΤΕΧΝΕΊΟ
ΣΧΟΛΉ ΗΛΕΚΤΡΟΛΌΓΩΝ ΜΗΧΑΝΙΚΏΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΗΜΑΤΩΝ, ΕΛΕΓΧΟΥ ΚΑΙ ΡΟΜΠΟΤΙΚΗΣ

Ανάπτυξη Προγράμματος Ανάλυσης Συχνοτήτων και Υπολογισμού Ευαισθησιών σε Περιβάλλον MATLAB

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Ι. Δρυς

Επιβλέπων : Νικόλαος Γ. Μαράτος

Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 21η Οκτωβρίου 2015.

.....	.....	.....
Ν.Μαράτος	Κ.Σ. Τζαφέστας	Χ.Ψυλλάκης
Καθηγητής Ε.Μ.Π.	Επίκ. Καθηγητής Ε.Μ.Π.	Λέκτορας Ε.Μ.Π.

Αθήνα, Οκτώβριος 2015

.....

Δρυς Νίκος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται την ανάπτυξη ενός προγράμματος ανάλυσης κυκλωμάτων στη συχνότητα και υπολογισμού ευαισθησιών, σε περιβάλλον MATLAB. Αυτό που το διακρίνει σε σχέση με άλλα προγράμματα ανάλυσης κυκλωμάτων είναι ότι δίνει τη δυνατότητα στο χρήστη να εισάγει καινούριες τιμές σε επιλεγμένα στοιχεία μέσα σε ένα κύκλωμα που τον ενδιαφέρει, χωρίς να χρειάζεται να επαναλαμβάνει όλα τα χαρακτηριστικά του κυκλώματος. Το πρόγραμμα αυτό ονομάζεται Driplomatiki και μπορεί να αποτελέσει ένα ιδιαίτερα χρήσιμο εργαλείο στα χέρια ένος αναλυτή ή/και σχεδιαστή κυκλωμάτων, αφού αλλάζοντας τιμές σε συγκεκριμένα στοιχεία του κυκλώματος, ο χρήστης, έχει τη δυνατότητα να εφαρμόσει επαναληπτικούς αλγόριθμους σε αυτά ώστε να βελτιστοποιήσει την απόκριση ενός κυκλώματος στη συχνότητα, ή/και να μετριάσει την ευαισθησία κάποιων τάσεων ή ρευμάτων σε συγκεκριμένα στοιχεία.

Λέξεις Κλειδιά

Τροποποιημένη Μέθοδος Κόμβων, Ανάλυση Συχνοτήτων με MATLAB, Υπολογισμός Ευαισθησιών με MATLAB, Ανάλυση Κυκλωμάτων με MATLAB, Ανάλυση Κυκλωμάτων

Abstract

This paper presents a program that performs frequency analysis and sensitivity computations in circuits. It is distinguishable from other circuit analyzing programs because it gives the user the ability to input new values for some chosen elements of a circuit without calling for a restatment of the features of the circuit (topology, values of the invariable elements etc). This program is developed in MATLAB environment and is called Driplomatiki. It could be a useful tool in the hands of a circuit analyser or/and designer since, having the capability of changing the values of specific elements, he can optimize the frequency response of a circuit, or/and moderate the sensitivity of specific voltages or currents with respect to some given elements.

Keywords

Modified Nodal Analysis, Frequency Analysis with MATLAB, Computation of Sensitivities with MATLAB, Circuit Analysis with MATLAB, Circuit Analysis

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε την περίοδο Μαϊος 2015 – Οκτώβριος 2015 στη σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών ΕΜΠ με αρκετό κόπο και προσωπική εργασία.

Θα ήθελα να ευχαριστήσω, πάνω από όλους, τον καθηγητή μου κύριο Νικόλαο Μαράτο, που με εμπιστεύτηκε για την εργασία και μου παρείχε τις πολύτιμες γνώσεις του και την βοήθειά του σε οποιοδήποτε πρόβλημα αντιμετώπισα. Η καθοδήγησή του υπήρξε καταπληκτική και η εν γένει συνεργασία μας καθ’ όλη τη διάρκεια εκπόνησης της εργασίας ήταν άψογη.

Επίσης θέλω να ευχαριστήσω τον πατέρα μου Ιωάννη, τη μητέρα μου Μαρία και την οικογένεια μου συνολικά καθώς στέκονται πάντα δίπλα μου. Τους φίλους μου και τις φίλες μου για όλα αυτά που έζησα μαζί τους και κατάφεραν να χρωματίσουν τα φοιτητικά μου χρόνια με έντονες πινελιές. Τέλος, θέλω να ευχαριστήσω όλους αυτούς που κατάφεραν να με πείσουν ότι η κατάκτηση της γνώσης είναι μια αέναη και πολύτιμη διαδικασία.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Η ανάλυση κυκλωμάτων είναι ένας τομέας που έχει απασχολήσει και συνεχίζει να απασχολεί την βιομηχανική, επιστημονική και πανεπιστημιακή κοινότητα. Με την ανάπτυξη των ηλεκτρονικών υπολογιστών αρκετά εργαλεία ανάλυσης κυκλωμάτων, όπως για παράδειγμα το SPICE, έχουν σχεδιαστεί ώστε να μπορεί να γίνεται με γρήγορο και αυτόματο τρόπο η ανάλυση των τάσεων, των ρευμάτων και άλλων μεγεθών σε ένα κύκλωμα. Παράλληλα, πολλά πακέτα λογισμικού που παρέχουν ισχυρά μαθηματικά εργαλεία, όπως το *MATLAB, Simulink* κ.α. έχουν παραχθεί και αναπτυχθεί με σκοπό να έχουν ευρεία εφαρμογή σε ερευνητικά και μή ζητήματα.

Το MATLAB είναι ένα λογισμικό αριθμητικών μεθόδων που γίνεται αυξανόμενα δημοφιλές στην επιστημονική και την πανεπιστημιακή κοινότητα αφού παρέχει εύχρηστες συναρτήσεις, απλή αναπαράσταση διαφόρων μαθηματικών ποσοτήτων (μητρών, διανυσμάτων κ.λ.π.), απλή προγραμματιστική λογική, δυνατότητα γραφής επιπρόσθετων συναρτήσεων και εκτεταμένες εργαλειοθήκες για πληθώρα χρήσεων σε διάφορους επιστημονικούς τομείς. Ας σημειωθεί όμως ότι το MATLAB δεν περιέχει, όπως είναι σήμερα, εργαλειοθήκη για ανάλυση κυκλωμάτων.

Η παρούσα διπλωματική εργασία πραγματεύεται την ανάπτυξη ενός προγράμματος ανάλυσης κυκλωμάτων στη συχνότητα και υπολογισμού ευαισθησιών των εξόδων, σε περιβάλλον MATLAB. Το πρόγραμμα αυτό, εν ονόματι Driplomatiki, όχι μόνο έχει σκοπό να βοηθήσει έναν αναλυτή, ή εναν σχεδιαστή κυκλωμάτων να μάθει πληροφορίες για ένα κύκλωμα που τον ενδιαφέρει, αλλά δίνει και τη δυνατότητα σε κάποιον χρήστη να αλλάζει τις τιμές σε συγκεκριμένα στοιχεία του κυκλώματος, (τα οποία θα ονομάζονται σχεδιάσιμα) ενώ άλλα να τα διατηρεί σταθερά (μη-σχεδιάσιμα). Αυτό διευκολύνει την περιγραφή ενός κυκλώματος, πλεονεκτεί στην περίπτωση που το κύκλωμα χρειάζεται να αναλυθεί επανειλλημένα με διαφορετικές κάθε φορά τιμές στοιχειών, αλλά δίνει και τη δυνατότητα σε έναν προγραμματιστή και

ταυτόχρονα σχεδιαστή/αναλυτή κυκλωμάτων να υλοποιήσει κάποιον επαναληπτικό αλγόριθμο βελτιστοποίησης ως προς τις τιμές επιλεγμένων στοιχείων του κυκλώματος.

Στο συγκεκριμένο πρόγραμμα η τοπολογία του κυκλώματος, υπό ανάλυση, θεωρείται γνωστή και δίνεται ως είσοδος από τον χρήστη, μέσω της γραφής ενός απλού αρχείου εν ονόματι InputPart.m το οποίο αποτελεί κομβικό σημείο για το πρόγραμμα. Είσοδοι του προγράμματος Driplomatiki θεωρούνται και οι τιμές των σχεδιάσιμων στοιχείων, τα οποία έχει θεωρήσει ο χρήστης, αλλά και οι συχνότητες στις οποίες ζητείται να γίνει η ανάλυση.

Η κατάστρωση των απαραίτητων εξισώσεων του κυκλώματος και η τελική ανάλυση στη συχνότητα γίνεται με την Τροποποιημένη Μέθοδου των Κόμβων κάνοντας τη χρήση Δύο Γράφων (Modified Nodal Analysis using Two Graphs). Η θεωρητική ανάπτυξη της συγκεκριμένης μεθόδου αναλύεται στο Κεφάλαιο 2. Το πρόγραμμα που αναπτύχθηκε υπολογίζει επίσης τις ευαισθησίες των εξόδων ως προς τα σχεδιάσιμα στοιχεία κάνοντας χρήση της μεθόδου του Συζυγούς Συστήματος, όπως αυτή αναλύεται στο Κεφάλαιο 3.

Η έξοδος του προγράμματος είναι 2 μήτρες που περιέχουν τις τιμές των εξόδων του κυκλώματος και των ευαισθησιών τους, ως προς τα σχεδιάσιμα στοιχεία καθώς και τα διαγράμματα Bode των εξόδων του κυκλώματος, για τις τιμές συχνοτήτων που έλαβε το πρόγραμμα Driplomatiki από τον χρήστη. Έξοδοι, επίσης, μπορούν να θεωρηθούν και άλλα ενδιαφέροντα διανύσματα που περιγράφονται αναλυτικά στο Κεφάλαιο 4 μαζί με τις οδηγίες χρήσης του προγράμματος.

Στο Κεφάλαιο 5 γίνεται χρήση του προγράμματος Driplomatiki ώστε να επαληθευτεί η ορθή του εκτέλεση. Τέλος, στο Παράρτημα παρατίθεται ο κώδικας του συγκεκριμένου προγράμματος.

Κεφάλαιο 2

Ανάλυση γραμμικών κυκλωμάτων με την τροποποιημένη μέθοδο των κόμβων με δύο γράφους (ΤΜΚ2Γ).

Πριν προχωρήσουμε στον τρόπο που λειτουργεί το πρόγραμμα Driplomatiki, το οποίο χρησιμοποιεί την τροποποιημένη μέθοδο των κόμβων με δύο γράφους στο MATLAB, θα ήταν χρήσιμο και βολικό να αναφέρουμε κάποιες βασικές έννοιες της τροποποιημένης μεθόδου των κόμβων με δύο γράφους ώστε να εξοικειωθεί ο αναγνώστης με το θεωρητιθκό υπόβαθρο που χρησιμοποιεί η παρούσα διπλωματική εργασία.

Α) Η απλή μέθοδος των κόμβων

Ας θεωρήσουμε πρώτα ένα κύκλωμα με n+1 κόμβους και b κλάδους καθώς επίσης ένα γράφο G να είναι ο προσανατολισμένος γράφος του κυκλώματος.

Ορισμός 1.1

Μήτρα προσπτώσεως A_α του γράφου G είναι η μήτρα (n+1)xb διαστάσεων της οποίας τα στοιχεία συμπληρώνονται ως εξής:

α_{ij} = 1, αν ο κλάδος j αναχωρεί από τον κόμβο i

α_{ij} = -1, αν ο κλάδος j προσπίπτει στον κόμβο i

α_{ij} = 0, αν ο κλάδος j δεν ενώνεται με τον κόμβο j

Οι γραμμές της A_α αντιστοιχούν στους κόμβους της G και οι στήλες στους κλάδους. Κάθε στήλη της A_α έχει ακριβώς δύο μη-μηδενικά στοιχεία, όπου το ένα ισούται με 1 και το άλλο με -1.

Ορισμός 1.2

Ελαττωμένη μήτρα προσπτώτσεως A του γράφου G είναι η μήτρα nxb διαστάσεων που προκύπτει από την A_α με αφαίρεση της γραμμής που αντιστοιχεί στον κόμβο αναφοράς (συμβατικώς αναφερόμενος ως "γή").

Με βάση τους παραπάνω ορισμούς, αν θεωρήσουμε διάνυσμα ρευμάτων στους κλάδους $I = [i_\alpha , i_\beta , ... , i_b]^T$, επαληθεύεται ότι ο νόμος ρευμάτων Kirchoff μπορεί να γραφτεί σε μητρική μορφή ως n+1 γραμμικά εξαρτημένες εξισώσεις

$$A_\alpha I = 0$$

ή σε n γραμμικά ανεξάρτητες εξισώσεις (αν αφαιρεθεί ο κόμβος "γη" και ο γράφος G είναι συνεκτικός)

$$A I = 0$$

Πέραν του διανύσματος ρευμάτων, είναι χρήσιμο να θεωρήσουμε και τα διανύσματα τάσεων κλάδων $V_b=[V_\alpha , V_\beta , ... , V_b]^T$ και τάσεων κόμβων $V_n=[V_\alpha , V_\beta , ... , V_n]^T$.

Είναι εύκολο να διαπιστωθεί ότι μεταξύ των τάσεων κόμβων και των τάσεων κλάδων ισχύει η σχέση :

$$V_b = A^T V_n \quad (1)$$

η οποία εκφράζει σε μητρική μορφή το γεγονός ότι η τάση ενός κλάδου ισούται με τη διαφορά των τάσεων-προς-γή των κόμβων με τους οποίους συνδέεται ο κλάδος.

Στη μέθοδο των κόμβων είναι βασική η υπόθεση ότι κάθε κλάδος του γράφου G περιέχει: είτε μια ανεξάρτητη πηγή ρεύματος, είτε ένα γραμμικό στοιχείο που έχει περιγραφή αγωγιμότητας (δηλαδή που περιγράφεται από τη σχέση).

Περιγραφή αγωγιμότητας μπορεί να έχουν τα κάτωθι κυκλωματικά στοιχεία:

Ονομασία	*Περιγραφική*	*Τύπος*	*Αγωγιμότητα*

	Σχέση		
Αντίσταση	$\mathbf{i_k = v_k}$	*Παθητικό*	$\mathbf{y_k =}$
Πηνίο	$\mathbf{i_k = v_k}$, $\mathbf{i_k = v_k}$	*Παθητικό*	$\mathbf{y_k =}$ ή $\mathbf{y_k =}$
Πυκνωτής	$\mathbf{i_k} = j\omega C\mathbf{v_k}$, $\mathbf{i_k} = sC\mathbf{v_k}$	*Παθητικό*	$\mathbf{y_k} = j\omega C$ ή $\mathbf{y_k} = sC$
Ανεξάρτητη Πηγή Ρεύματος	$\mathbf{i_k = J}$		
Εξαρτημένη πηγή ρεύματος απο τάση (VCT)	$\mathbf{i_k} = g\mathbf{v_\lambda}$	*Ενεργητικό*	

Παθητικά λέγονται τα στοιχεία που περιγράφονται από τη γενική σχέση i = y v και y ονομάζεται η αγωγιμότητα του στοιχείου.

Σημαντική παρατήρηση : Οι ανεξάρτητες και εξαρτημένες πηγές τάσεως που ενδεχομένως υπάρχουν στο κύκλωμα είναι δυνατό να μετατραπούν σε ισοδύναμες ανεξάρτητες πηγές ρεύματος με χρήση των θεωρημάτων Thevenin-Norton.

Τελικώς, για την κατάστρωση των εξισώσεων των κόμβων υπάρχει μια βασική *προϋπόθεση* που σχετίζεται με τον τρόπο αρίθμησης των κλάδων του κυκλώματος: αριθμούνται πρώτα οι κλάδοι που αντιστοιχούν σε παθητικά στοιχεία ή ενεργητικά στοιχεία και τελευταίοι οι κλάδοι που αντιστοιχούν σε ανεξάρτητες πηγές ρεύματος. Έτσι τόσο η μήτρα A και τα διανύσματα I και V_b χωρίζονται με τον κάτωθι τρόπο:

A = , I = , V_b =

ο δείκτης p αναφέρεται στους κλάδους παθητικών στοιχείων ή εξαρτημένων πηγών και ο δείκτης J στους κλάδους ανεξάρτητων πηγών ρεύματος.

Με βάση τα παραπάνω ο Νόμος Ρευμάτων Kirchoff γράφεται :

Με τα νέα δεδομένα η σχέση (1) γράφεται ως εξής :

και οι σχέσεις των (επιτρεπόμενων) στοιχείων γράφονται σε μητρική μορφή :

όπου η μήτρα αγωγιμότητας κλάδων Y_b κατασκευάζεται με το κάθε στοιχείο να ακολουθεί τη σχέση $y_{κκ} = y_κ$ και τα υπόλοιπα της σειράς μηδενικά αν ο κλάδος που αντιστοιχεί στην σειρά αποτελείται από παθητικό στοιχείο ενώ αν είναι ενεργητικός συμπληρώνεται $y_{κλ} = g$ με τα υπόλοιπα της σειράς μηδενικά αν θεωρήσουμε ότι εξαρτάται από την τάση του κλάδου λ, με κέρδος g.

Καταληκτικά, οι εξισώσεις των κόμβων λαμβάνονται με απαλοιφή του I_p από τις σχέσεις (2) και (3) παίρνοντας :

όπου είναι η μήτρα αγωγιμότητος κόμβων (διαστάσεως nxn) και το διάνυσμα ανεξαρτήτων πηγών εντάσεως κόμβων (μήκους n).

Για ανάλυση συνεχούς ρεύματος η μήτρα αγωγιμότητος κόμβων και τα διανύσματα V_n και είναι πραγματικοί αριθμοί, ενώ για ανάλυση Ημιτονικής Μόνιμης Κατάστασης είναι μιγαδικοί αριθμοί που εξαρτώνται από την συχνότητα *ω.*

Β) Η τροποιημένη μέθοδος των κόμβων με χρήση δύο γράφων

Η Τροποποιημένη Μέθοδος των Κόμβων με Δύο Γράφους (ΤΜΚ2Γ) είναι μια επέκταση της μεθόδου των κόμβων με σκοπό να έχει πιο γενική εφαρμογή. Για να γίνει κάτι τέτοιο η ΤΜΚ2Γ περιλαμβάνει άμεσα στη διατύπωση της τα εξής στοιχεία:

- ανεξάρτητες πηγές τάσης
- στοιχεία που έχουν περιγραφική σχέση σε μορφή αντιστάσεως V= Zi
- εξαρτημένες πηγές τάσης ελεγχόμενες από τάση ή ρεύμα v=μv',v=ri'
- γενικά δίθυρα στοιχεία με περιγραφική σχέση: YV + ZI = W

Με την προηγούμενη νοοτροπία και συμπεριλαμβάνοντας τα παραπάνω στοιχεία, η ΤΜΚ2Γ καλύπτει τις παραπάνω απαιτήσεις που μπορεί να έχουν κάποιες εφαρμογές, διατηρώντας συγχρόνως σε μεγάλο βαθμό την απλότητα της μεθόδου των κόμβων.

Για την διατύπωση της θεωρούμε, αρχικά, όλες τις άγνωστες τάσεις και τα ρεύματα και τις μεταξύ τους σχέσεις, μέσω των εξισώσεων Kirchoff ή μέσα από τις περιγραφικές σχέσεις των στοιχείων. Ας είναι I_b και V_b οι άγνωστες μεταβλητές του κυκλώματος. Από τις προαναφερθείσες σχέσεις παίρνουμε ένα σύστημα εξισώσεων της μορφής:

όπου I είναι η μοναδιαία μήτρα bxb διαστάσεων και $W_b = 0$ για περιγραφές αντιστάσεων και αγωγιμοτήτων.

Η ΤΜΚ2Γ εφορμεί απο τρείς απαλοιφές που μπορούν να γίνουν σε αυτό το σύστημα.

1) Αντικατάσταση των $V_b = A^T V_n$
2) Χωρισμός των περιγραφικών σχέσεων των στοιχείων
    a.i) Γενικά δίθυρα στοιχεία : $Y_1V_1+Z_1I_1 = W_1$
    a.ii) Στοιχεία με περιγραφή αγωγιμότητας $Y_2V_2 = I_2$
    a.iii) ανεξάρτητες πηγές ρευμάτων : $I_3 = J$

και με αντίστοιχο χωρισμό της μήτρας πρόσπτωσης A οδηγούμαστε στην απαλοιφή των ρευμάτων I_2 και στην μεταφορά των I_3 στο δεξιό μέλος του συστήματος. Καταλήγουμε σε σχέσεις της μορφής:

3) Πολλά από τα ρεύματα του διανύσματος I_1 και πολλές από τις τάσεις κλάδων που εμπεριέχονται στο V_b και άρα στο V_n είναι ίσες με μηδέν ή δεν ενδιαφέρει η

γνώση τους. Αυτές οι "αδιάφορες" τάσεις και ρεύματα μπορούν να απαλειφθούν με τη βοήθεια δύο γράφων. Στον έναν θα προκύψουν οι άγνωστες τάσεις (V-γράφος) και στον άλλον θα προκύψουν τα άγνωστα ρεύματα (I-γράφος). Ο νόμος ρευμάτων Kirchoff μπορεί να εκφρασθεί με βάση τον I-γράφο και ο νόμος τάσεων με βάση τον V-γράφο.

Για τη σχεδίαση του I-γράφου οι κλάδοι οι οποίοι διαρρέονται από ρεύμα που δεν υπεσέρχεται στην περιγραφική σχέση του στοιχείου που περιέχουν και δεν ενδιαφέρει η γνώση τους βραχυκυκλώνονται. Οι κλάδοι που διαρρέονται από μηδενικό ρεύμα ανοιχτοκυκλώνονται.

Για τη σχεδίαση του V-γράφου, ένας κλάδος βραχυκυκλώνεται εαν η τάση στα άκρα του είναι μηδέν και ανοιχτοκυκλώνεται όταν η τάση στα άκρα του δεν συμπεριλαμβάνεται στην περιγραφική σχέση που περιέχει ο κλάδος και δεν ενδιαφέρει η γνώση της.

Αφού σχεδιαστούν οι δύο γράφοι του κυκλώματος χωρίζουμε τους κλάδους του κυκλώματος ανάλογα με τις περιγραφικές σχέσεις των στοιχείων που περιέχουν:

- Στοιχεία με περιγραφή αγωγιμότητας : $I_1 = Y_1V_1$
- Στοιχεία με περιγραφή αντιστάσεως : $V_2 = Z_2I_2$
- Ανεξάρτητες πηγές ρεύματος $I_3 = J$
- Ανεξάρτητες πηγές τάσεως $V_3 = E$
- Εξαρτημένες πηγές ρεύματος από ρεύμα $I_4 = \alpha I_5$
- Εξαρτημένες πηγές τάσεως από τάση $V_4 = \mu V_5$
- Γενικά δίθυρα στοιχεία $Y_6V_6 + Z_6I_6 = W$

Με αντίστοιχο τρόπο χωρίζονται οι μήτρες προσπτώσεως στον κάθε γράφο. Αν πριν είχαμε τη μήτρα A_I του I-γράφου και την A_V του V-γράφου τώρα θεωρούμε αντίστοιχα τις και για κ = 1,2,...,6.

Για τις ανάγκες του κεφαλαίου αυτού, θα συμβολίσουμε τις τάσεις των κόμβων του V-γράφου ως προς γή ως .

Οι εξισώσεις της ΤΜΚ2Γ προκύπτουν με αντικατάσταση των I_1, I_3, I_4 και της στους νόμους ρευμάτων Kirchoff ( ) και των $V_κ$ από τους νόμους τάσεων Kirchoff στις περιγραφικές σχέσεις των στοιχείων.

Τελικώς καταλήγουμε στην επίσημη σχέση της ΤΜΚ2Γ σε μητρική μορφή :

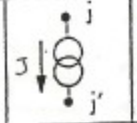
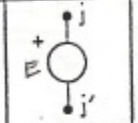
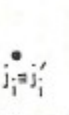
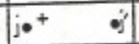
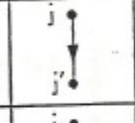
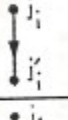
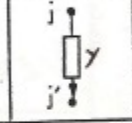
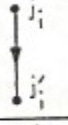
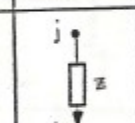
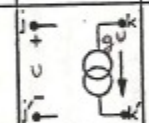
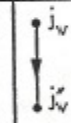
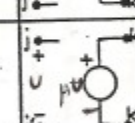
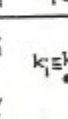
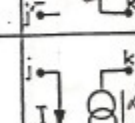
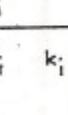
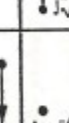
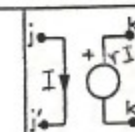
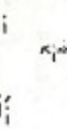
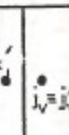
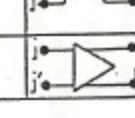
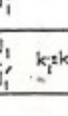
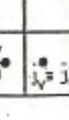
Αξίζει να σημειώσουμε εδώ ότι οι εξισώσεις της ΤΜΚ2Γ δεν είναι απαραίτητο να καταστρωθούν από εκτέλεση πολλαπλασιασμών μητρών και εύρεση των μητρών προσπτώσεως του κάθε γράφου. Η μήτρα αυτή μπορεί απλούστερα να συμπληρωθεί με *επαναληπτική προσθήκη της συμβολής του κάθε κλάδου* κάνοντας απλή χρήση των δύο γράφων σύμφωνα με τον Πίνακα 1. Το πρόγραμμα Driplomatiki χρησιμοποιεί αυτή τη μέθοδο συμπλήρωσης των εξισώσεων αφού προσφέρει ευνοϊκότερο τρόπο για την προγραμματιστική υλοποίηση.

Έτσι, η κατασκευή των I και V γράφων γίνεται επαναληπτικά, από τον γράφο του κυκλώματος, θεωρώντας την επίδραση του κάθε κλάδου ξεχωριστά. Οι συμβολές των παθητικών στοιχείων με περιγραφικές σχέσεις αγωγιμότηταςκαθώς και των εξαρτημένων πηγών ρεύματος από τάση γράφονται επαναληπτικά, όπως στη μέθοδο των κόμβων, σε μια μήτρα διαστάσεων n_I x n_V όπου n_I είναι ο αριθμός των κόμβων του I-γράφου και n_V ο αριθμός των κόμβων του V-γράφου. Η μήτρα αυτή είναι το τμήμα των παραπάνω εξισώσεων. Το τμήμα συμπληρώνεται από τη συμβολή των ανεξάρτητων πηγών ρεύματος αφού τοποθετηθούν στο σταθερό διάνυσμα του δεξιού μέλους.

Οι συμβολές όλων των στοιχείων φαίνονται στον Πίνακα 1. Καθένα από τα υπόλοιπα στοιχεία προσθέτει είτε μια γραμμή είτε μια στήλη (μεταβλητή I) είτε μια γραμμή και μια στήλη (μια εξίσωση και μια μεταβλητή I) στην επαναληπτικά κατασκευαζόμενη μήτρα ή/και στο σταθερό διάνυσμα του δεξιού μέλους.

Στο τέλος της διαδικασίας προκύπτει η μήτρα και το σταθερό διάνυσμα της ΤΜΚ2Γ.

Πίνακας 1. Ιδεατά στοιχεία στην τροποποιημένη μέθοδο των κόμβων με δύο γράφους.

ΣΤΟΙΧΕΙΟ	ΣΥΜΒΟΛΟ	I-ΓΡΑΦΟΣ	V-ΓΡΑΦΟΣ	ΜΗΤΡΑ
Ανεξάρτητη πηγή έντασης				$j_k \begin{bmatrix} -J \\ J \end{bmatrix}$
Ανεξάρτητη πηγή τάσης				$m+1 \begin{bmatrix} \cdots & \cdots \\ 1 & -1 \end{bmatrix}, \begin{bmatrix} - \\ E \end{bmatrix}$
Ανοικτοκύκλωμα				—
Βραχυκύκλωμα				$\begin{matrix} & I \\ j_k & \\ j_{k'} & \end{matrix} \begin{bmatrix} & \vdots & 1 \\ & \vdots & -1 \end{bmatrix}$
Σύνθετη αγωγιμότητα				$\begin{matrix} & j_v & j_{v'} \\ j_k & & \\ j_{k'} & & \end{matrix} \begin{bmatrix} y & -y \\ -y & y \end{bmatrix}$
Σύνθετη Αντίσταση				$\begin{matrix} & j_v & j_{v'} & I \\ j_k & & & \\ j_{k'} & & & \\ m+1 & & & \end{matrix} \begin{bmatrix} & & \vdots & 1 \\ & & \vdots & -1 \\ 1 & -1 & \vdots & -z \end{bmatrix}$
Μετατροπέας τάσης σε ρεύμα				$\begin{matrix} & j_v & j_{v'} \\ k_k & & \\ k_{k'} & & \end{matrix} \begin{bmatrix} g & -g \\ -g & g \end{bmatrix}$
Μετατροπέας τάσης σε τάση				$\begin{bmatrix} & & & \\ & & & \\ -\mu & \mu & 1 & -1 \end{bmatrix}$
Μετατροπέας ρεύματος σε ρεύμα				$\begin{matrix} & & & I \\ j_k & & & \\ j_{k'} & & & \\ k_k & & & \\ k_{k'} & & & \end{matrix} \begin{bmatrix} & & \vdots & 1 \\ & & \vdots & -1 \\ & & \vdots & \alpha \\ & & \vdots & -\alpha \end{bmatrix}$
Μετατροπέας ρεύματος σε τάση				$\begin{matrix} & & & I \\ j_k & & & \\ j_{k'} & & & \\ m+1 & & & \end{matrix} \begin{bmatrix} & & \vdots & 1 \\ & & \vdots & -1 \\ 1 & -1 & \vdots & -r \end{bmatrix}$
Τελεστικός Ενισχυτής				—

Κεφάλαιο 3

Δομή και συνοπτική περιγραφή του προγράμματος Driplomatiki

Αφού συνοψίσαμε τη θεωρία που αποτελεί τη βάση για να πραγματοποιηθεί αυτή η διπλωματική εργασία, θα εξηγήσουμε σε αύτο το κεφάλαιο ποιός είναι ο τρόπος λειτουργίας του προγράμματος και πως καταλήγει στον στόχο του να αναλύσει ένα κύκλωμα στη συχνότητα, να υπολογίσει τις ευαισθησίες ως προς συγκεκριμένα στοιχεία του κυκλώματος και να αποτελέσει ένα πρόσθετο εργαλείο στα χέρια ενός σχεδιαστή κυκλωμάτων.

Α) Η Δομή και ο τρόπος λειτουργίας του προγράμματος Driplomatiki

Το πρόγραμμα Driplomatiki είναι ένα πρόγραμμα ανάλυσης κυκλωμάτων στη συχνότητα και υπολογισμού ευαισθησιών εξόδου ως προς συγκεκριμένα στοιχεία, τα οποία επιθυμεί ο χρήστης. Τα στοιχεία του κυκλώματος διακρίνονται σε σχεδιάσιμα και μη-σχεδιάσιμα. Οι τιμές των μη-σχεδιάσιμων στοιχείων διατηρούνται σταθερές, σε διαδοχικές εκτελέσεις του προγράμματος, ενώ οι τιμές των σχεδιάσιμων στοιχείων είναι δυνατό να αλλάζουν μεταξύ διαδοχικών εκτελέσεων.

Αρχικά, για την εκτέλεση του προγράμματος, απαιτούνται από τον χρήστη τα συγκεκριμένα στοιχεία:

- Τοπολογία του κυκλώματος που ζητείται να αναλυθεί.
- Διάκριση στοιχείων σε σχεδιάσιμα και μη-σχεδιάσιμα.
- Τιμές των μη-σχεδιάσιμων στοιχείων και ονόματα των σχεδιάσιμων.
- Τιμές των συχνοτήτων στις οποίες θέλει ο χρήστης να γίνει ανάλυση.

Μόλις συλλέξει αυτά τα στοιχεία, το συγκεκριμένο πρόγραμμα πραγματοποιεί τα παρακάτω:

- Παράγει τους I-V Γράφους του δεδομένου κυκλώματος.

- Καταστρώνονται οι εξισώσεις της ΤΜΚ2Γ.
- Γίνεται ανάλυση συχνότητας του κυκλώματος και υπολογίζονται οι έξοδοι.
- Κατασκευάζονται τα διαγράμματα Bode.
- Υπολογίζει τις ευαισθησίες των εξόδων, ως προς τα σχεδιάσιμα στοιχεία που έχει δώσει ο χρήστης.

Το πρόγραμμα Driplomatiki παίρνει ως είσοδο την τοπολογία ενός κυκλώματος, η οποία λειτουργεί και ως γράφος του κυκλώματος, και αρχικά παράγει τους Ι και V γράφους του κυκλώματος με τον τρόπο που περιγράφει ο Πίνακας 1. Γίνεται διαχωρισμός των στοιχείων του κυκλώματος σε σχεδιάσιμα και μή σχεδιάσιμα, ώστε να μπορεί ο χρήστης του προγράμματος να αλλάζει τιμές σε κάποια στοιχεία ενώ άλλα να παραμένουν σταθερά. Ο χρήστης καλείται να πληκτρολογήσει στο πρόγραμμα τις αρχικές τιμές των σχεδιάσιμων στοιχείων που έχει ορίσει, έτσι ώστε να γίνουν οι κατάλληλοι υπολογισμοί. Έτσι, χωρίς να επανεισάγει κάθε φορά όλες τις τιμές των στοιχείων του κυκλώματος ελέγχει την ευαισθησία μόνο για συγκεκριμένα στοιχεία που τον ενδιαφέρουν.

Στο πρόγραμμα Driplomatiki θεωρείται ότι οι είσοδοι του κυκλώματος που αναλύει είναι ανεξάρτητες πηγές ρεύματος ή τάσης ενώ η έξοδος ή οι έξοδοι του κυκλώματος είναι τάσεις ανοιχτοκύκλωσεως ή ρεύματα βραχυκυκλώσεως.

Στη συνέχεια, κατασκευάζεται η μήτρα Τ και το πρόγραμμα παίρνει ως είσοδο τις μπάντες συχνοτήτων στις οποίες ο χρήστης θέλει να γίνει ανάλυση. Από τους Ι και V γράφους, που έλαβε από το πρώτο κομμάτι κώδικα, με τρόπο που θα αναλυθεί πλήρως στις παραγράφους Β και Γ, ευρίσκεται η καθυστέρηση ομάδας της κάθε εξόδου καθώς και η ευαισθησία της σε σχέση με κάθε σχεδιάσιμο στοιχείο. Ο υπολογισμός ευαισθησιών γίνεται επαναληπτικά για την κάθε συχνότητα.

Τέλος, γίνονται τα διαγράμματα Bode και απεικονίζονται τα αποτελέσματα της ανάλυσης ευαισθησιών.

Β) Εισαγωγή τοπολογίας και στοιχείων του κυκλώματος

Η τοπολογία του κυκλώματος στο πρόγραμμα είσαγεται με χρήση του αρχείου InputPart.m. Αυτό γίνεται με τρόπο απλό, αφού ο χρήστης όταν θελήσει να εισάγει κάποιο καινούριο στοιχείο στο κύκλωμα (οποιοδήποτε στοιχείο που αναφέρει ο Πίνακας 1) πληκτρολογεί μια καινούρια σειρά στο αρχείο InputPart.m με το όνομα του είδους του στοιχείου, τους κόμβους στους οποίους βρίσκεται και την τιμή του, αν θέλει να είναι μή-σχεδιάσιμο, ή το όνομα του μέσα σε εισαγωγικά, αν θέλει να είναι σχεδιάσιμο.

Τα στοιχεία που υποστηρίζει το πρόγραμμα με τα ονόματα τους είναι τα εξής:

- e για ανεξάρτητη πηγή τάσης
- cs για ανεξάρτητη πηγή ρεύματος
- vtov για εξαρτημένη πηγή τάσης από τάση
- jtov για εξαρτημένη πηγή τάσης από ρεύμα
- jtoj για εξαρτημένη πηγή ρεύματος από ρεύμα
- vtoj για εξαρτημένη πηγή ρεύματος από τάση
- opamp για τελεστικό ενισχυτή
- g για γενικευμένη αγωγιμότητα
- z για γενικευμένη αντίσταση
- c για πυκνωτή
- r για αντίσταση
- l για πηνίο
- oc για ανοιχτοκύκλωμα
- sc για βραχυκύκλωμα

Έτσι, για παράδειγμα αν θέλει ο χρήστης να περιγράψει ένα κύκλωμα που περιέχει μια αγωγιμότητα από τον κόμβο 1 στον κόμβο 0, με τιμή 100 Ω^{-1} , έναν

πυκνωτή με τιμή 10 μF από τον 1 στον 2 και μια ανεξάρτητη πηγή τάσης από τον κόμβο 1 στον κόμβο 0, σχεδιάσιμη, το αρχείο InputPart.m θα πρέπει να είναι το

```
function InputPart(~)

g(0,1,100);
c(1,2,0.00001);
e(1,2,'e1');

end
```

Ως είσοδοι του κυκλώματος θεωρούνται οι ανεξάρτητες πηγές τάσης ή έντασης που δίνει ο χρήστης. Δηλαδή, αν ο χρήστης θέλει να εισάγει μια είσοδο π.χ. τάσεως σε ένα κύκλωμα, απλά προσθέτει μια ανεξάρτητη πηγή τάσεως στους επιθυμητούς κόμβους. Αν, τώρα, θελήσει να λάβει ως έξοδο μια τάση μεταξύ δύο κόμβων, (π.χ. πάνω σε μια αντίσταση) αρκεί να θεωρήσει μια τάση ανοιχτοκύκλωσης σε αυτούς τους κόμβους και να της δώσει ένα όνομα. Σε άλλη περίπτωση, αν χρειαστεί να μετράται το ρεύμα που διαρέει έναν κλάδο, τότε ο χρήστης πρέπει να θεωρήσει δύο κόμβους βραχυκυκλωμένους στη μια άκρη του κλαδου, αντί για έναν, και να δώσει ένα όνομα στο ρεύμα που τους διαρέει.

Για να δώσουμε ποια έξοδο θέλουμε να έχει το κύκλωμα ή, γενικότερα, αν θέλουμε να μετρήσουμε κάπου την τάση ή το ρεύμα, αρκεί να τοποθετήσουμε ένα βραχυκύκλωμα (για ρεύματα) ή ενα ανοιχτοκύκλωμα (για τάσεις) στην περιοχή κώδικα του InputPart.m μαζί με το επιθυμητό όνομα σε εισαγωγικά. Έτσι, αν θέλει ο χρήστης στο παραπάνω κύκλωμα να μετρήσει την τάση από τον κόμβο 2 στη γή, αρκεί να τροποιήσει τον παραπάνω κώδικα ως εξής:

```
function InputPart(~)

g(0,1,100);
c(1,2,0.00001);
e(1,0,'e1');
oc(2,0,'Vout');

end
```

Η λογική με την οποία το πρόγραμμα κατασκευάζει και αντιλαμβάνεται τον γράφο του κυκλώματος, κάθε φορά που ο χρήστης τοποθετεί ένα στοιχείο στο κύκλωμα, είναι η λογική του *Πίνακα Πρόσπτωσης (Incidence matrix)*. Κατ'αυτήν τη νοοτροπία η τοπολογία ενός κυκλώματος αναπαρίσταται στον Η/Υ σαν ένας πίνακας 4xb, όπου b ο αριθμός των κλάδων, του οποίου οι στήλες αντιστοιχούν στους κλάδους του κυκλώματος και οι γραμμές περιέχουν τις εξής πληροφορίες για τους κλάδους:

1. Είδος του στοιχείου που περιέχεται στον κλάδο (π.χ. 'G' για αγωγιμότητα)
2. Είτε η τιμή του στοιχείου, είτε το όνομα του, αν είναι σχεδιάσιμο
3. Κόμβος αναχωρήσεως κλάδου
4. Κόμβος προσπτώσεως κλάδου
5. Αύξων αριθμός ταυτότητας που ενημερώνει αν είναι σταθερό ή σχεδιάσιμο το στοιχείο (0 για σταθερά, ένας αύξων αριθμός για σχεδιάσιμα)

Έτσι, αν δώσουμε την παραπάνω τοπολογία στο κύκλωμα και πληκτρολογήσουμε στο terminal του MATLAB τη λέξη MyGraph, για να μάθουμε τον γράφο του κυκλώματος, θα πάρουμε τον πίνακα Προσπτώσεως:

MyGraph =

'Type'	'G'	'C'	'V'	'OC'
'Value'	[100]	[1.0000e-05]	[10]	'Vout'
'K1'	[0]	[1]	[1]	[2]
'K2'	[1]	[2]	[0]	[0]
'ID'	[0]	[0]	[1]	[0]

Αξίζει να σημειωθεί ότι σε περίπτωση που ο χρήστης δώσει κάποιους αριθμούς κόμβων και, τελικά, οι αριθμοί δεν ακολουθούν την σειρά των φυσικών αριθμών (δηλ. 0,1,2,3,...) το πρόγραμμα αυτόματα επανονομάζει τους κόμβους ώστε να υπάρχει, στο κύκλωμα, η σωστή αρίθμηση κόμβων για την περαιτέρω σωστή ανάλυση του.

Επιπλέον, είναι αξιοσημείωτη η διαχείριση του προγράμματος σχετικά με τους κόμβους του κανονικού γράφου οι οποίοι αντιστοιχίζονται κατάλληλα με τους κόμβους του I και V γράφου ώστε να παραχθούν οι τελευταίοι με σωστό τρόπο.

Για το παραπάνω κύκλωμα, για παράδειγμα, πρέπει να γίνουν οι σωστές αντιστοιχήσεις όπως δείχνουν οι παρακατω "χάρτες" αντιστοίχησης του προγράμματος:

```
>> IMap

IMap =

    [0]    [1x2 double]
    [1]    [         2]

>> VMap

VMap =

    [0]    [0]
    [1]    [1]
    [2]    [2]
```

ώστε να παραχθούν επιτυχώς οι παρακάτω I και V-γράφοι του κυκλώματος:

```
>> IGraph

IGraph =

    'Type'     'C'
    'Value'    [1.0000e-05]
    'K1'       [          0]
    'K2'       [          1]
    'ID'       [          0]

>> VGraph

VGraph =

    'Type'     'G'      'C'              'V'      'OC'
    'Value'    [100]    [1.0000e-05]     [10]     'Vout'
    'K1'       [  0]    [          1]    [ 1]     [   2]
    'K2'       [  1]    [          2]    [ 0]     [   0]
    'ID'       [  0]    [          0]    [ 1]     [   0]
```

Γ) Ανάλυση του κυκλώματος και υπολογισμός των εξόδων

i) Η έξοδος

Οι εξισώσεις που έχουμε τελικά, ύστερα από την ανάλυση που έγινε στο Κεφάλαιο 2, αφορούν τις τάσεις και τα ρεύματα όλων των κλάδων ενός γραμμικού κυκλώματος. Τα μεγέθη αυτά ευρίσκονται από την επίλυση του παρακάτω συστήματος γραμμικών εξισώσεων:

Το διάνυσμα περιέχει τις τάσεις των κόμβων του V-γράφου και ενδεχομένως ρεύματα κλάδων τα οποία λαμβάνουν σημαντικό ρόλο στις εξισώσεις του κυκλώματος. Το δίανυσμα , διαστάσεων bx1, περιλαμβάνει τις ανεξάρτητες πηγές του κυκλώματος.

Η έξοδος φ του κυκλώματος είναι, συνήθως, μια τάση ή ένα ρεύμα κλάδου του κυκλώματος. Επομένως, η φ ισούται με ένα στοιχείο του διανύσματος ή με τη διαφορά δύο στοιχείων του διανύσματος , αφού αυτό περιέχει τάσεις κόμβων και, ενδεχομένως, ρεύματα κλάδων. Μπορεί, λοιπόν, να γραφεί στη μορφή:

όπου είναι ένα σταθερό διάνυσμα e_k της μοναδιαίας μήτρας ή ίσο με τη διαφορά e_k - $e_{k'}$ δύο στηλών της μοναδιαίας μήτρας. Εάν έχουμε παραπάνω από μια έξοδους ϕ_i, i = 1,2,...,n , το δίανυσμα μετατρέπεται σε μήτρα D μεγέθους nxb (όπου b ο αριθμός των κλάδων) της οποίας η κάθε στήλη είναι το διάνυσμα $_i$, μήκους b, που αντιστοιχεί στην έξοδο ϕ_i. Προφανώς ισχύει

Αξίζει να αναφέρουμε εδώ ότι η μήτρα και το διάνυσμα (κατ'επέκταση και η φ) είναι μιγαδικές συναρτήσεις της συχνότητας *s* ή *jω*.

Η εύρεση των εξόδων πραγματοποιείται, από τις παραπάνω σχέσεις, με χρήση των μαθηματικών εργαλείων του MATLAB για συστήματα γραμμικών εξισώσεων. Πρώτα ευρίσκεται το διάνυσμα και ύστερα το διάνυσμα φ.

ii) Διαγράμματα Bode

Για την ανάλυση στη συχνότητα, ο χρήστης καλείται να φτιάξει το διάνυσμα των τιμών των συχνοτήτων στις οποίες θα γίνει η ανάλυση. Αυτό γίνεται πληκτρολογώντας τα δύο άκρα ενός εύρους συχνοτήτων (αφού προφανώς ερωτηθεί από το πρόγραμμα) και τον αριθμό των δειγμάτων που θέλει να πάρει από αυτό το εύρος, αφού του ζητηθεί από το πρόγραμμα. Η διαδικασία της εισαγωγής επαναλαμβάνεται, και αν ο χρήστης εισάγει παραπάνω από ένα εύρος, τότε κατασκευάζεται ένα κοινό διάνυσμα με όλες τις τιμές συχνοτήτων, στις οποίες γίνεται η ανάλυση.

Από την έξοδο φ, και αφού υπάρχει πρόσθετο εσωτερικό διάνυσμα του προγράμματος που περιέχει τις πληροφορίες για τις εισόδους, παίρνουμε το διάγραμμα Bode της κάθε εξόδου ως προς την κάθε είσοδο. Προφανώς για να κατασκευαστέι το διάγραμμα Bode μιας εξόδου ϕ_i ως προς μια είσοδο λαμβάνουμε στον έναν άξονα τις συχνότητες, σε λογαριθμική κλίμακα, ενώ στον άλλο τις τιμές της συνάρτησης:

Δ) Ανάλυση ευαισθησιών με τη μέθοδο του συζυγούς ή ανάστροφου συστήματος

Η μέθοδος με την οποία το πρόγραμμα Driplomatikh βρίσκει τις ευαισθησίες, ονομάζεται *μέθοδος του συζυγούς συστήματος*. Για να εξηγήσουμε στον αναγνώστη πως προκύπτουν οι ευαισθησίες των εξόδων ϕ_i, (i = 1,2,...,n) ως προς συγκεκριμένα στοιχεία, πρέπει να αναλυθεί ο τρόπος που προκύπτει η μέθοδος του συζυγούς συστήματος.

Η ευαισθησία της εξόδου , ως προς κάποιο στοιχείο x_i, i = 1,2,...,m του κυκλώματος, είναι η μερική παράγωγος . Με παραγώγιση της εξίσωσης (4) ως πρός x_i μας δίνει:

Από την εξίσωση (5) προκύπτει:

Έστω ότι

(7)

Το τελευταίο σύστημα γραμμικών εξισώσεων αποτελεί το συζυγές σύστημα εξισώσεων και για την κάθε έξοδο ισχύει:

Είναι σημαντικό να αναφέρουμε ότι η επίλυση των συστημάτων (4) και (7) πραγματοποιείται με τη βοήθεια της παραγοντοποίησης LU που παρέχει το πρόγραμμα MATLAB.

Αντικαθιστώντας την (7) στην (6) λαμβάνουμε την τελική μορφή:

Βλέπουμε τώρα ότι μέσω της επίλυσης του συζυγούς συστήματος απλοποιείται η εύρεση των ευαισθησιών του κυκλώματος.. Η αναπαράσταση του συστήματος (8) δίνει σημαντικά πλεονεκτήματα, όπως το γεγονός ότι το διάνυσμα W είναι αραιό, αφού γνωρίζουμε ότι συνήθως τα κυκλώματα δεν περιλαμβάνουν πάνω από μία με δύο διεγέρσεις. Άρα γνωρίζουμε εξαρχής ότι κάποιοι υπολογισμοί θα μειωθούν. Τα δύο πιο σημαντικά χαρακτηριστικά του συστήματος (8) είναι ότι:

- Για υπολογισμούς ευαισθησιών ως προς πηγή τάσης ή ρεύματος το κομμάτι που αντιστοιχεί στο είναι 0, ενώ η μερική παράγωγος είναι της μορφής και η εξίσωση (8) απλοποιείται στην εξίσωση :

- Το διάνυσμα , συνήθως, δεν εξαρτάται από τα στοιχεία ως προς τα οποία υπολογίζουμε ευαισθησίες, επομένως η μερική παράγωγος είναι 0. Επιπλέον, στην ΤΜΚ2Γ κάθε στοιχείο κάνει μέχρι 4 καταχωρήσεις στην

μήτρα . Αν θεωρήσουμε οτι i,j είναι οι δείκτες που αφορούν τον κλάδο

ελέγχου και τα k,l αφορούν τον κλάδο πηγής τότε οι καταχωρήσεις αυτές

μπορούν να γραφούν στη μορφή:

όπου = 0 για αντιστάσεις και = 1 για πυκνωτές και πηνία. Η μερική παραγώγιση τους ως προς δίνει

Επομένως, η ευαισθησία μιας εξόδου ϕ_i ως προς ένα στοιχείο δίνεται από την απλοποιημένη μορφή:

Τα δύο αυτά χαρακτηριστικά συμπληρώνουν τη μέθοδο του συζυγούς συστήματος, η οποία αρχικά, συνιστά την εύρεση των διανυσμάτων και και

ύστερα καθιστά τους υπολογισμούς των ευαισθησιών ιδιαίτερα απλούς, χρησιμοποιώντας τις εξισώσεις (9) και (10).

Για την εύρεση της ευαισθησίας ως προς τη συχνότητα κάνουμε την

παρατήρηση ότι το διάνυσμα είναι ανεξάρτητο της συχνότητας και άρα η

μερική παράγωγος έχει αμιγώς φανταστικό μέρος. Επομένως, εάν

θεωρήσουμε ότι η μήτρα Τ γράφεται στη μορφή : η ευαισθησία της εξόδου

ως προς τη συχνότητα δίνεται από την εξίσωση:

Αυτός ο τύπος για την ευαισθησία ως προς τη συχνότητα έχει χρησιμοποιηθεί από το πρόγραμμα Driplomatiki για την εύρεση των ευαισθησιών της εξόδου ως προς τη συχνότητα. Μια άλλη μορφή του ίδιου τύπου, που ενδείκνυται για υπολογισμούς με το χέρι, είναι η παρακάτω:

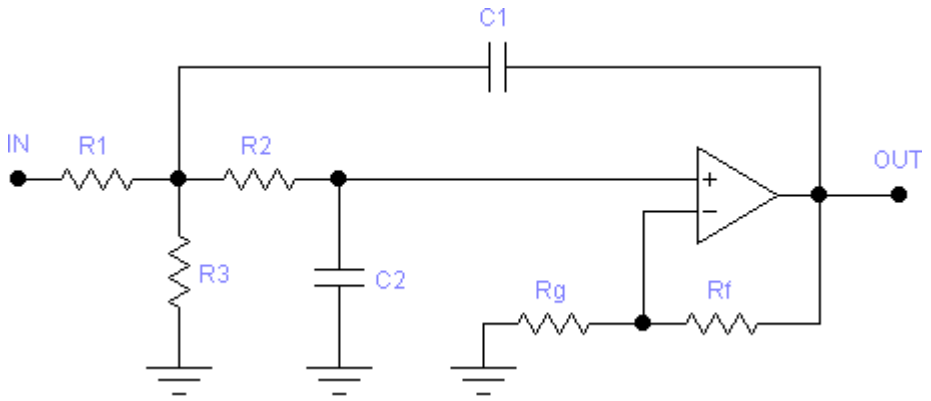
Τέλος η καθυστέρηση ομάδας, που είναι ένα μέγεθος που μας δίνει αρκετή πληροφορία για ένα κύκλωμα, δίνεται από τη σχέση:

Κεφάλαιο 4

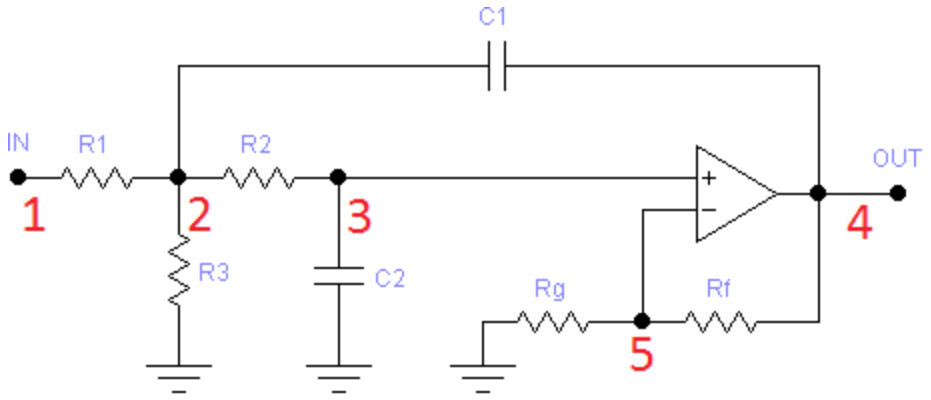
Οδηγίες χρήσης για το πρόγραμμα Driplomatiki

Για να ανοίξει ένα οποιοδήποτε αρχείο κώδικα (όπως το πρόγραμμα Driplomatiki) στο πρόγραμμα MATLAB αρκεί να κάνουμε κλίκ στην επιλογή Open και να το ανοίξουμε, αφού προφανώς εντοπίσουμε σε ποία διεύθυνση (directory) είναι αποθηκευμένο στον υπολογιστή. Ύστερα κάνουμε κλίκ στην επιλογή "Run" και το πρόγραμμα τρέχει.

Αρχικά, ο χρήστης πρέπει να εισάγει στο πρόγραμμα την τοπολογία του κυκλώματος το οποίο θέλει να εξετάσει. Όπως εξηγήθηκε και πριν, κάτι τέτοιο θα γίνει με την τροποποίηση του αρχείου **InputPart.m** πληκτρολογώντας σειρά - σειρά τα στοιχεία που θέλουμε να έχει το πρόγραμμα. Για παράδειγμα αν ζητήσει ο χρήστης να κάνει ανάλυση στο παρακάτω κύκλωμα:



θα πρέπει να αριθμήσει τους κόμβους, με όποια σειρά θέλει. Παρακάτω παραθέτουμε ένα παράδειγμα αρίθμησης κόμβων:



Στη συνέχεια, θα πρέπει να εισάγει τα στοιχεία ένα-ένα, γράφοντας πρώτα τον τύπο του στοιχείου που θέλει (στο Κεφάλαιο 2 παρ. Β κατονομάζονται οι επιτρεπόμενοι τύποι στοιχείων με τα ονόματα τους) και ύστερα τον κόμβο αναχωρήσεως και προσπτώσεως. Επίσης, πρέπει να δώσει την τιμή του (αν θέλει να είναι σταθερό) ή το όνομα του, μέσα σε εισαγωγικά (αν θέλει να είναι σχεδιάσιμο). Τέλος, για να πάρει έξοδο ή εξόδους από το κυκλώμα θα πρέπει να θεωρήσει ανοιχτοκύκλωμα (για μέτρηση τάσης) ή βραχυκύκλωμα (για μέτρηση ρεύματος) και ύστερα να δώσει όνομα. Για την παραπάνω τοπολογία κυκλώματος το αρχείο InputPart.m θα πρέπει να μοιάζει με το παρακάτω:

```
function InputPart(~)
```

```
e(1,0, 'e1');
g(1,2,20);
g(2,0,0.5);
g(2,3,10);
c(3,0,0.0001);
c(2,4,0.001);
opamp(3,5,4,0);
g(5,0,30);
g(5,4,25);
oc(4,0,'Vout');

end
```

Οι τιμές που δόθηκαν στο κύκλωμα είναι τυχαίες. Παρατηρούμε ότι μόνο η τάση εισόδου είναι σχεδιάσιμο στοιχείο στο κύκλωμα.

Το πρόγραμμα Driplomatiki ζητάει δύο πληροφορίες από το χρήστη για να κάνει την ανάλυση του:

α)Τις τιμές των σχεδιάσιμων στοιχείων και

β)Τις μπάντες συχνοτήτων για τις οποίες θα γίνει η ανάλυση.

Αυτό πραγματοποιείται με εύκολο τρόπο αφού το πρόγραμμα, όταν ξεκινήσει με πίεση της επιλογής "Run", ζητείται από το χρήστη να δώσει τιμές για το κάθε στοιχείο το οποίο του υποδεικνύει, γραπτώς.

Αφού πληκτρολογηθούν οι τιμές για τα σχεδιάσιμα στοιχεία και αποθηκευτεί το αρχείο ως *InputPart.m*, αρχίζει μια επαναληπτική διαδικασία εισαγωγής τιμών συχνοτήτων στη μορφή εύρους τιμών. Αν ο χρήστης θελει να εισάγει ένα καινούριο εύρος συχνοτήτων πληκτρολογεί πρώτα μια ελάχιστη συχνότητα Wmin, μια μέγιστη συχνότητα του εύρους Wmax, και τον αριθμό των δειγμάτων, Nsamples, για τα οποία θέλει να γίνουν υπολογισμοί, μέσα σε αυτό το εύρος.

Αυτό γίνεται επαναληπτικά, για όλες τις μπάντες συχνοτήτων που επιθυμεί ο χρήστης, μέσω ενός μενού επιλογών. Έτσι, το πρόγραμμα αντιλαμβάνεται ένα εύρος συχνοτήτων (Wmin,Wmax,Nsamples) με διακριτό τρόπο. Αν π.χ. δώσουμε το εύρος (10,20) hz και πληκτρολογήσουμε ότι θέλουμε 5 δείγματα, κατασκευάζεται, από το σύστημα, ένα διάνυσμα με τιμές [10 , 12.5, 15, 17.5,20] και εισάγεται στο ολικό διάνυσμα τιμών των συχνοτήτων. Τέλος ταξινομείται το διάνυσμα και πραγματοποιείται η ανάλυση.

Τα αποτελέσματα που δίνει το πρόγραμμα στον χρήστη είναι τα διαγράμματα Bode όλων των έξόδων ϕ_i ως προς τις εισόδους w_i και μια εικόνα με όλες τις ευαισθησιές που υπολογίστηκαν, για την κάθε τιμή συχνότητας που βρίσκεται στο διάνυσμα συχνοτήτων. Αν θελήσει ο χρήστης να μάθει περισσότερες πληροφορίες για το κύκλωμα του (αυτές που μπορεί τελικά να δώσει η ΤΜΚ2Γ) μπορεί απλά να πληκτρολογήσει, στο terminal του MATLAB, τα ονόματα των μεταβλητών που τον ενδιαφέρουν. Παρακάτω αναφέρονται κάποιες χρήσιμες μεταβλητές, με τα ονόματα τους:

- T_freq, η μήτρα της ΤΜΚ2Γ με μεταβλητή τη συχνότητα ω

- WhoZ και z για πληροφορίες σχετικά με το διάνυσμα
- W για το διάνυσμα
- WhoFi και Fi για πληροφορίες σχετικά με τις εξόδους
- D για τη μήτρα
- MyGraph, ο γράφος του κυκλώματος
- IGraph, VGraph, οι γράφοι που κατασκευάστηκαν σύμφωνα με την ΤΜΚ2Γ

- DesignVector, πληροφορίες για τα σχεδιάσιμα στοιχεία και τις τιμές τους
- Table1 για τις αποκρίσεις των εξόδων ως προς τις εισόδους (από αυτό κατασκευάζονται τα διαγράμματα Bode
- Table2 για τις ευαισθησίες ως προς τα σχεδιάσιμα στοιχεία και την ευαισθησία ως προς την συχνότητα

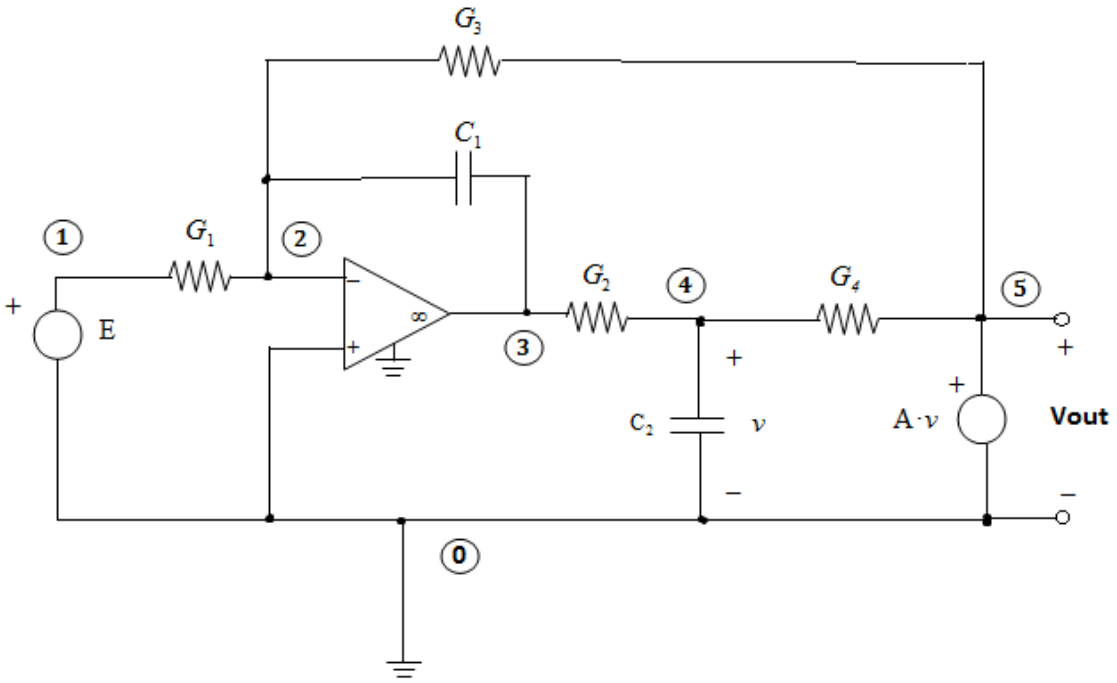
Κεφάλαιο 5

Πειραματική εκτέλεση του προγράμματος Driplomatiki και αποτελέσματα

Στη συνέχεια, θα χρησιμοποιήσουμε το πρόγραμμα Driplomatiki, με σκοπό να επιδείξουμε την χρησιμότητα του, για την επίλυση ενός πρόβληματος ανάλυσης συχνότητας και υπολογισμού ευαισθησιών. Θα παρουσιάζεται πρώτα η επισκοπική λύση του κάθε βήματος ανάλυσης με τα εργαλεία που μας παρέχει η ΤΜΚ2Γ και ύστερα θα αναγράφονται τα αποτελέσματα στο MATLAB, όπως αυτά υπολογίζονται και παρουσιάζονται από το πρόγραμμα Driplomatiki.

Α) Το κύκλωμα υπό ανάλυση

Η τοπολογιά του κύκλωματος που θα καλεστεί να αναλύσει το πρόγραμμα Driplomatiki, στο παρόν κεφάλαιο, φαίνεται στο παρακάτω σχήμα:



Για να εισαχθεί η τοπολογία του κυκλώματος στο πρόγραμμα πρέπει το αρχείο εισαγωγής InputPart.m να μοιάζει με το παρακάτω :

```
function InputPart(~)
e(1,0,10)
g(1,2,'g1')
c(2,3,'c1')
opamp(2,0,3,0)
g(2,5,'g3')
g(3,4,'g2')
c(4,0,'c2')
g(4,5,'g4')
vtov(4,0,5,0,'A')
oc(5,0,'Vout')

end
```

Υπενθυμίζουμε ότι αφού δώσαμε τα ονόματα των εξαρτημένων πηγών, των αντιστάσεων και των πυκνωτών, τα θεωρήσαμε σχεδιάσιμα στοιχεία, υπό την έννοια ότι κάθε φορά που εκτελείται το πρόγραμμα, μπορούμε να τους δίνουμε διαφορετικές τιμές. Η ανεξάρτητη πηγή θεωρήθηκε σταθερή και ίση με 10 V.Επίσης, με αυτόν τον τρόπο, δηλώσαμε ότι θέλουμε να ευρεθούν οι ευαισθησίες της εξόδου, ως προς αυτά τα στοιχεία.

Β) Εισαγωγή τιμών για τα σχεδιάσιμα στοιχεία και επιλογή συχνοτήτων ανάλυσης

Παρακάτω φαίνονται τα αντίστοιχα μηνύματα που δείχνει το terminal του MATLAB κάθε φορά που χρειάζεται είσοδο σε τιμές, κατά τη διάρκεια της εκτέλεσης. Χάριν διευκόλυνσης στους υπολογισμούς θα δώσουμε μοναδιαίες τιμές στις αγωγιμότητες και τους πυκνωτές:

```
Give Input for :
g1
1
Give Input for :
c1
1
```

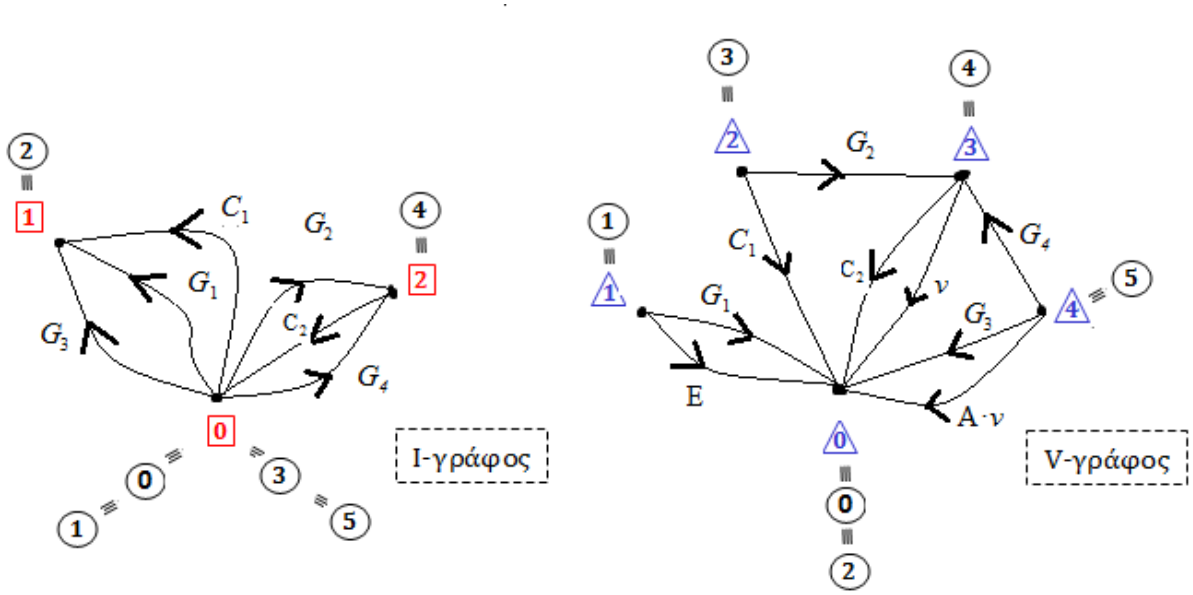
```
Give Input for :
g3
1
Give Input for :
g2
1


Give Input for :
c2
1
Give Input for :
g4
1
Give Input for :
A
4
```

επίσης, για να είναι ευανάγνωστη η μήτρα μήτρα Τ, θα εισάγουμε τη συχνότητα 1 . Παρακάτω φαίνεται πως δείχνει το τερματικό του MATLAB στη συγκεκριμένη φάση εκτέλεσης:

```
Give frequency ranges\n
You have two choices: \n
1.
Input (Wmin,Wmax,Nsamples)\n
2.
I have finished with input
Your choice:1
Wmin:1
Wmax:1
Number of samples:1
You have two choices: \n
1.
Input (Wmin,Wmax,Nsamples)\n
2.
I have finished with input
Your choice:2
```

Γ) I-V γράφοι του κυκλώματος, σύμφωνα με την ΤΜΚ2Γ και τον Πίνακα 1

Ο I και V γράφος το κυκλώματος που προαναφέρθηκε μπορεί να εξαχθεί επισκοπικά και με χρήση του Πίνακα 1:



Για να δούμε τα αντίστοιχα αποτελέσματα του προγράμματος Driplomatiki και για να επιβεβαιώσουμε ότι είναι σωστά παραθέτουμε την εικόνα του τερματικού, ζητώντας να μάθουμε τις τιμές των αντίστοιχων μεταβλητών:

```
>> IGraph

IGraph =

    'Type'     'G'    'C'    'G'    'G'    'C'    'G'
    'Value'    [1]    [1]    [1]    [1]    [1]    [1]
    'K1'       [0]    [1]    [1]    [0]    [2]    [2]
    'K2'       [1]    [0]    [0]    [2]    [0]    [0]
    'ID'       [2]    [3]    [4]    [5]    [6]    [7]

>> VGraph

VGraph =

    'Type'     'V'     'G'    'C'    'G'    'G'    'C'    'G'    'mIn'    'mOut'    'OC'
    'Value'    [10]    [1]    [1]    [1]    [1]    [1]    [1]    '-'      [   4]    'Vout'
    'K1'       [ 1]    [1]    [0]    [0]    [2]    [3]    [3]    [  3]    [   4]    [   4]
    'K2'       [ 0]    [0]    [2]    [4]    [3]    [0]    [4]    [  0]    [   0]    [   0]
    'ID'       [ 1]    [2]    [3]    [4]    [5]    [6]    [7]    [  0]    [   8]    [   0]
```

Υπενθυμίζουμε ότι στην αναπαράσταση ενός γράφου του κυκλώματος στο πρόγραμμα Driplomatiki η τελευταία σειρά 'ID' αντιστοιχεί σε έναν αριθμό του στοιχείου, ο οποίος

- αν είναι 0 τότε το στοιχείο είναι μη-σχεδιάσιμο
- αν είναι φυσικός τότε το στοιχείο είναι σχεδιάσιμο και ο αριθμός αυτός είναι ο αύξων αριθμός που του αντιστοιχεί.

Γ) Οι εξισώσεις της ΤΜΚ2Γ σε μητρική μορφή και οι ευαισθησίες τις εξόδου

Σύμφωνα με την ΤΜΚ2Γ και τον Πίνακα 1 μπορούν να καταστρωθούν, με το χέρι, οι γραμμικές εξισώσεις του κυκλώματος σε μητρική μορφή:

$$
\begin{array}{c}
\\
1 \\
2 \\
\text{E} \\
\text{A}\cdot\nu
\end{array}
\begin{array}{c}
\begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\
\begin{bmatrix}
-G_1 & -sC_1 & 0 & -G_3 \\
0 & -G_2 & G_2 + G_4 + sC_2 & -G_4 \\
0 & 0 & -A & 1 \\
1 & 0 & 0 & 0
\end{bmatrix}
\end{array}
\begin{bmatrix} \tilde{V}_1 \\ \tilde{V}_2 \\ \tilde{V}_3 \\ \tilde{V}_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ E \\ 0 \end{bmatrix}
$$

Η ορθή εκτέλεση των υπολογισμών στο πρόγραμμα, για τις τιμές των στοιχείων που δώσαμε προηγουμένως, επαληθεύεται, πληκτρολογώντας στο terminal του MATLAB τα ονόματα των μεταβλητών:

```
>>  T_freq

T_freq =

[ -1, -w*i,        0, -1]
[  0,    -1, w*i + 2, -1]
[  0,     0,       -4,  1]
[  1,     0,        0,  0]


>> WhoZ

WhoZ =

 V1
 V2
 V3
 V4


>> W

W =

     0
     0
     0
     10
```

Έπειτα, για να ελέγξουμε τους υπολογισμούς του προγράμματος επιλύουμε το σύστημα με το χέρι (με τη βοήθεια της μεθόδου Cramer) και λαμβάνουμε τις παρακάτω τιμές για τα z:

Επομένως:

Ενώ για την επίλυση του ανάστροφου συστήματος θα χρησιμοποιήσουμε την εξίσωση (7) :

Το παραπάνω σύστημα γραμμικών εξισώσεων θα επιλυθεί επίσης με Cramer:

Άρα :

Επαληθεύουμε ότι το πρόγραμμα Driplomatiki δίνει τα ίδια αποτελέσματα:

```
>> z

z =

  10.0000 + 0.0000i
   6.1538 + 0.7692i
  -9.2308 - 6.1538i
  -2.3077 - 1.5385i


>> z_Adj

z_Adj =

  -0.9231 - 0.6154i
  -0.6154 + 0.9231i
  -0.5385 + 0.3077i
  -0.9231 - 0.6154i
```

Έχουμε επίσης για την έξοδο:

Οι πληροφορίες που έχει αρχικώς το MATLAB για την έξοδο είναι το όνομα και το διάνυσμα d στην παρακάτω μορφή:

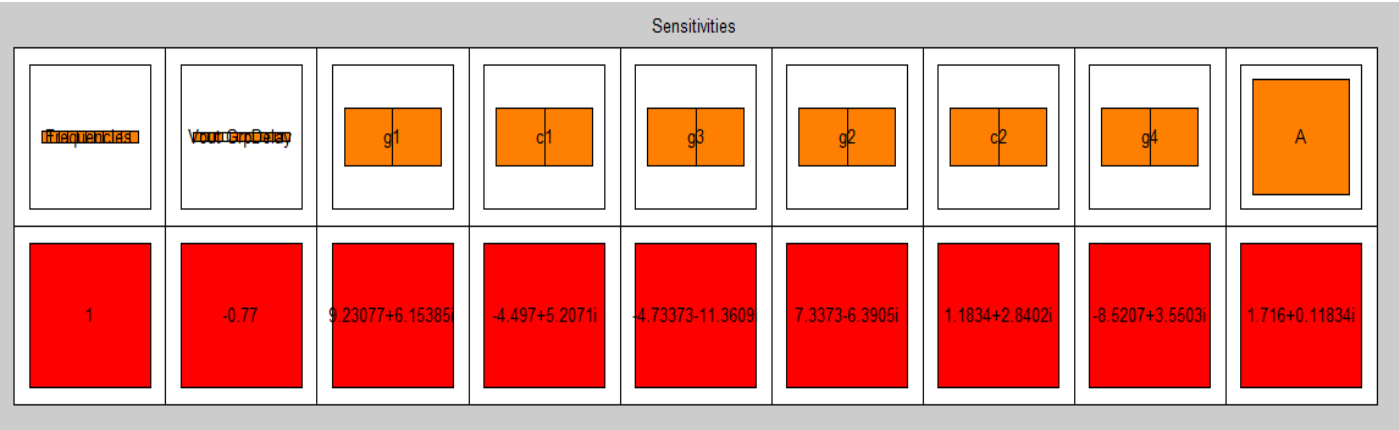
```
>> D

D =

     0     0     0     1


>> WhoFi

WhoFi =

Vout
```

Οι ευαισθησίες ως προς το κάθε στοιχείο, ως προς τη συχνότητα και η καθυστέρηση ομάδας, θα υπολογιστούν αρχικά με το χέρι (για λόγους ελέγχου του προγράμματος) με τη βοήθεια των εξισώσεων (9),(10),(11) και (12) του Κεφαλαίου 3.

Από τις εξισώσεις (11) και (12) θα πάρουμε την ευαισθησία της εξόδου ως προς τη συχνότητα:

και την καθυστέρηση ομάδας:

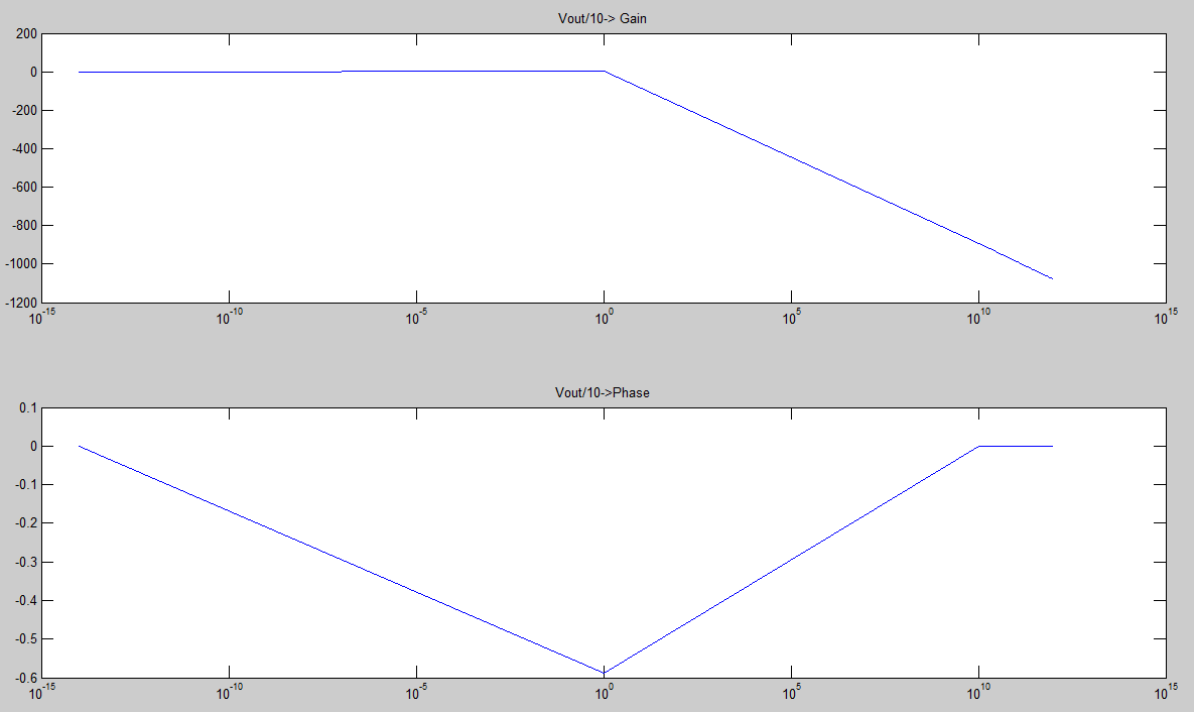
Παρακάτω είναι οι ευαισθησίες που δίνει το MATLAB στον χρήστη, αφού εκτελεστεί το πρόγραμμα Driplomatiki:



Παρατηρούμε ότι οι τιμές συμφωνούν.

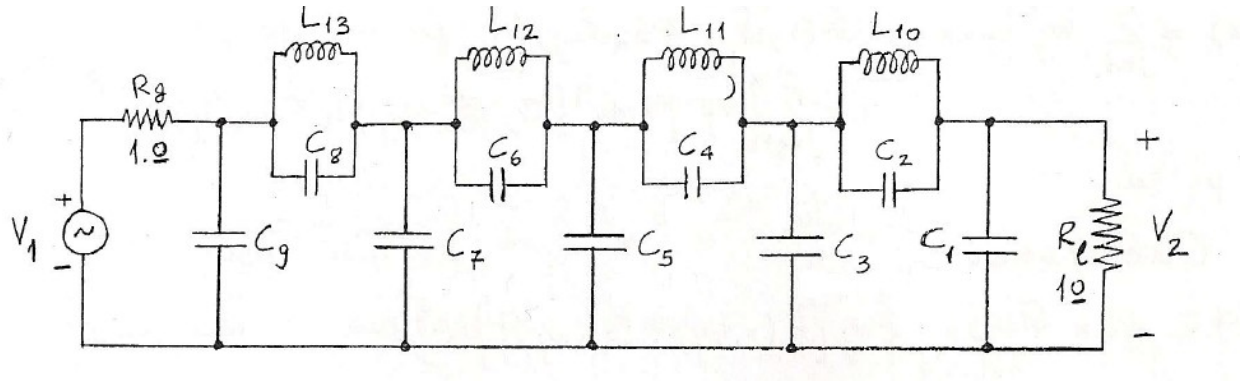
Δ) Διαγράμματα Bode

Παρακάτω δίδονται τα διαγράμματα Bode, του κυκλώματος που αναλύεται, στην περιοχή (. Λαμβάνονται 100 δείγματα σε αυτήν την περιοχή:



Προφανώς φαίνεται ότι το κύκλωμα υπο ανάλυση ήταν ένα βαθυπερατό φίλτρο.

Εκτος από το παραπάνω παράδειγμα, το πρόγραμμα Driplomatiki έχει δοκιμασθεί επιτυχώς σε πιο πολύπλοκα αλλά και σε πιο μεγάλα κυκλώματα. Ένα απο αυτά είναι το βαθυπερατό φίλτρο του σχήματος:



το οποίο εισάγεται στο πρόγραμμα ως έτσι:

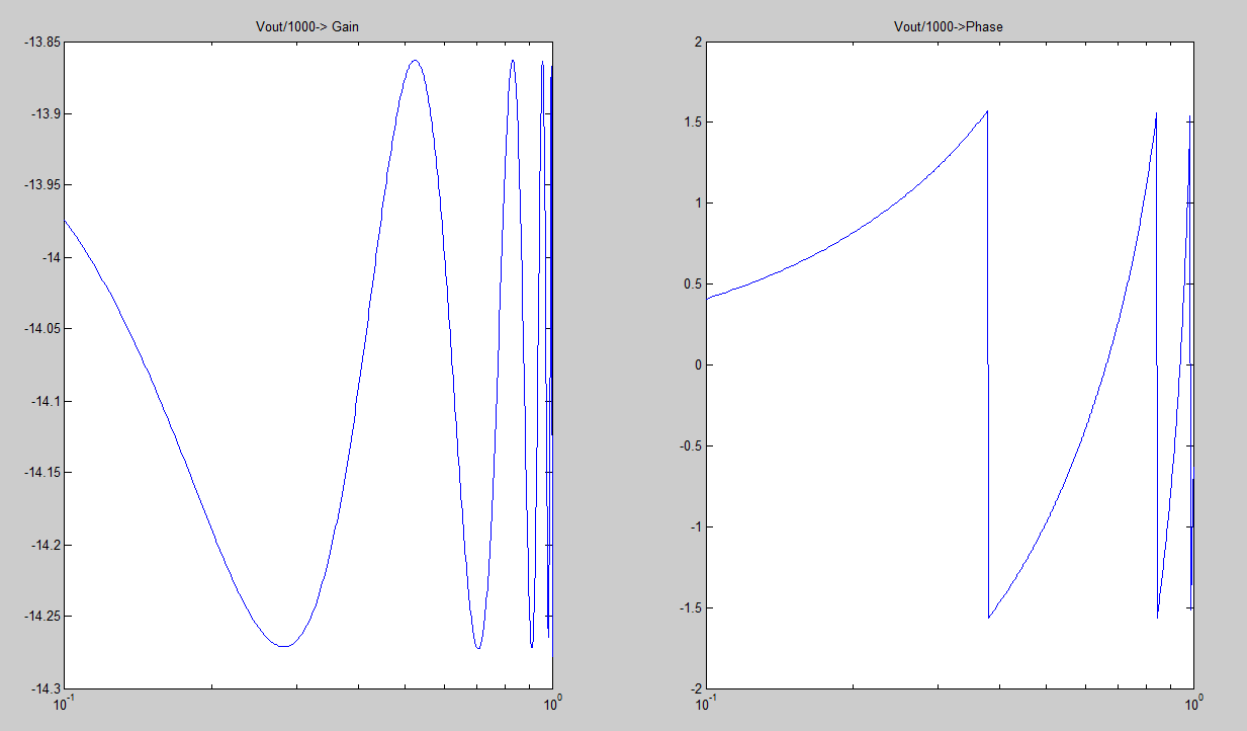
```
function InputPart(~)
e(1,0,10)
r(1,2,1)
l(2,3,1.192)
c(2,3,0.2078)
c(2,0,1.176)
c(3,0,1.401)
c(3,4,1.382)
l(3,4,0.5979)
c(4,0,0.8233)
l(4,5,0.4457)
c(4,5,2.013)
c(5,0,1.018)
l(5,6,0.7503)
c(5,6,0.8304)
c(6,0,0.7875)
r(6,0,1)
oc(6,0,'Vout')
end
```

Παρατηρούμε ότι το συγκεκριμένο κύκλωμα είναι πολύπλοκο στην ανάλυση αφού η μήτρα Τ (δηλαδή οι εξισώσεις που το διέπουν) είναι η παρακάτω:

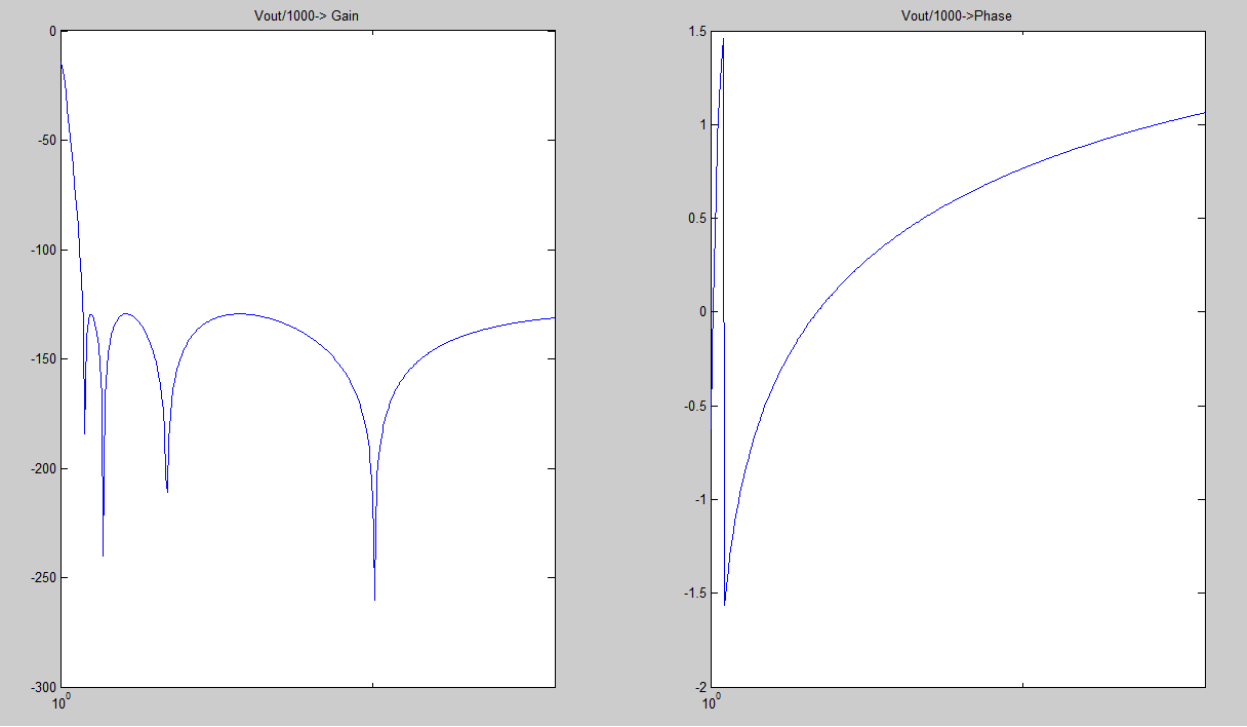
```
T_freq =

[ 0,  (w*6919*i)/5000, -(w*1039*i)/5000,              0,              0,              0, -1,  0,            1,              0,              0,              0]
[ 0, -(w*1039*i)/5000,  (w*7477*i)/2500,   -(w*691*i)/500,              0,              0,  0,  0,           -1,              1,              0,              0]
[ 0,                0,   -(w*691*i)/500, (w*42183*i)/10000, -(w*2013*i)/1000,              0,  0,  0,            0,             -1,              1,              0]
[ 0,                0,                0, -(w*2013*i)/1000, (w*19307*i)/5000,   -(w*519*i)/625,  0,  0,            0,              0,             -1,              1]
[ 0,                0,                0,                0,   -(w*519*i)/625, (w*16179*i)/10000,  0,  1,            0,              0,              0,             -1]
[ 1,               -1,                0,                0,              0,              0, -1,  0,            0,              0,              0,              0]
[ 0,                0,                0,                0,              0,              1,  0, -1,            0,              0,              0,              0]
[ 0,                1,               -1,                0,              0,              0,  0,  0, -(w*149*i)/125,              0,              0,              0]
[ 0,                0,                1,               -1,              0,              0,  0,  0,            0, -(w*5979*i)/10000,              0,              0]
[ 0,                0,                0,                1,             -1,              0,  0,  0,            0,              0, -(w*4457*i)/10000,              0]
[ 0,                0,                0,                0,              1,             -1,  0,  0,            0,              0,              0, -(w*7503*i)/10000]
[ 1,                0,                0,                0,              0,              0,  0,  0,            0,              0,              0,              0]
```

Παρόλαυτά το πρόγραμμα Driplwmatiki για το εύρος συχνοτήτων (0.1,3) μας δίνει τα παρακάτω, σωστά, διάγραμματα Bode .

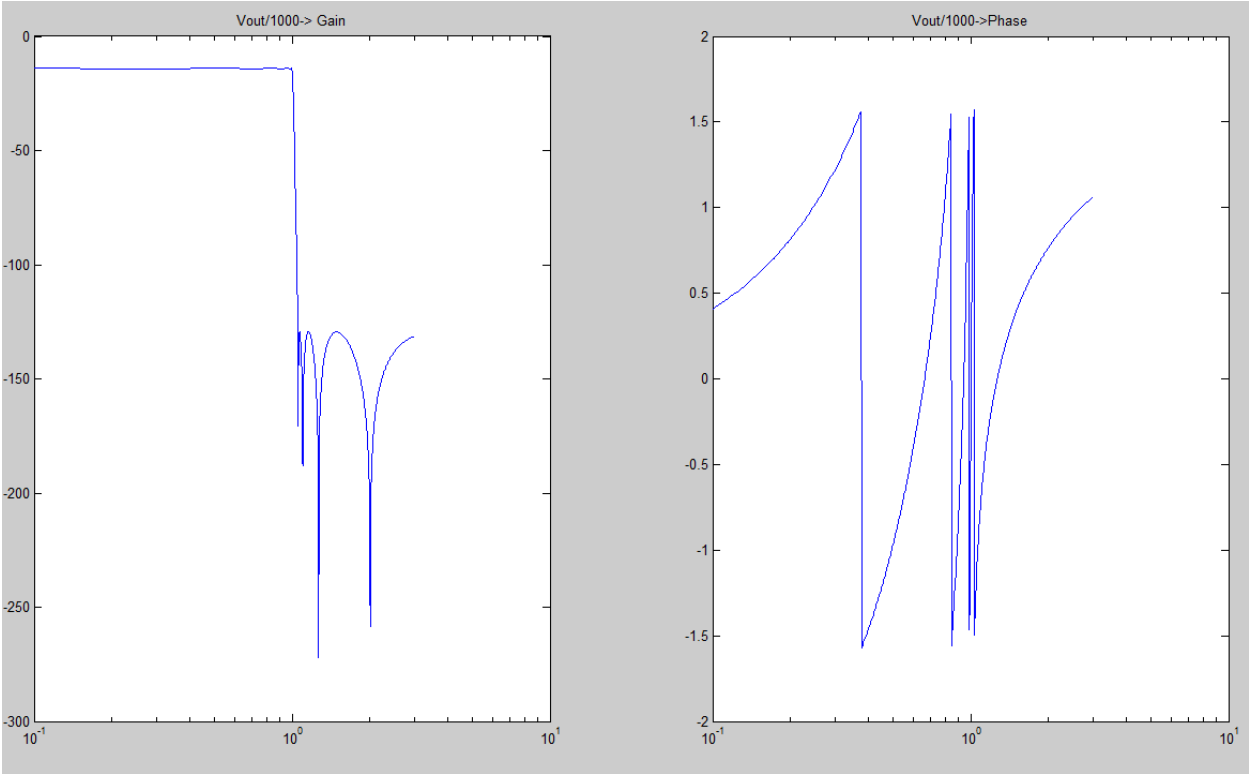
Διάγραμμα 1. Επιλεγμένο εύρος: (0.1,1) . 500 δείγματα.



Διάγραμμα 2. Επιλεγμένο εύρος: (1,3) . 500 δείγματα.


Vout/1000-> Gain
0
-50
-100
-150
-200
-250
-300
10^0
Vout/1000->Phase
1.5
1
0.5
0
-0.5
-1
-1.5
-2
10^0

Διάγραμμα 3. Επιλεγμένο εύρος: (0.1,3) . 500 δείγματα



Συμπεράσματα

Όπως παρουσιάστηκε και στα προηγούμενα Κεφάλαια και ιδιαίτερα στο Κεφάλαιο 5 το πρόγραμμα Driplomatiki που αναπτύχθηκε σε περιβάλλον MATLAB μπορεί να δώσει πολλές πληροφορίες για ένα κύκλωμα. Είναι ιδιαίτερα χρήσιμο και ισχυρό σε έμπειρα χέρια και μπορεί να αναλύσει επαρκώς κυκλώματα που ποικίλλουν σε μέγεθος και πολυπλοκότητα. Κάποιος που θέλει να αναλύσει στη συχνότητα ή/και να υπολογίσει ευαισθησίες κάποιου μεγέθους ως προς ένα άλλο, σε ένα κύκλωμα, μπορεί να το κάνει απλούστατα παίρνοντας τον κώδικα που βρίσκεται στο Παράρτημα. Με τον ίδιο τρόπο, ένας σχεδιάστης κυκλωμάτων μπορεί να εισάγει ένα κύκλωμα στο MATLAB και να βελτιστοποιήσει την απόκριση του κυκλώματος στη συχνότητα και την

ευαισθησία των εξόδων ως προς τις τιμές επιλεγμένων σχεδιάσιμων στοιχείων. Αυτό γίνεται, επαναληπτικά, με εισαγωγή τιμών στα σχεδιάσιμα στοιχεία και εξαγωγή των επιθυμητών αποκρίσεων ή/και ευαισθησιών.

ΠΑΡΑΡΤΗΜΑ : Κώδικας MATLAB

Παρακάτω αναφέρεται ο κώδικας MATLAB του προγράμματος Driplomatiki τις συναρτήσεις και τα σχόλια που περιέχει.

```
%Driplwmatiki

clear all;
global InitialGraph DesignVector DesIdx;
InitialGraph = {'Type';'Value';'K1';'K2';'ID'};
DesignVector = {'Name'; 'ID'; 'Value'; 'Type'};
DesIdx = 1;
%Input part serves as the input of the program
InputPart();
Frequencies = GiveFreqVals();
%InitialGraph now is an incidence matrix of the circuit
InitialGraph
DesignVector
%We make nodes a 2-by-Length matrix that has the mappings of the nodes from
```

```
%which we can take useful information
Length = size(InitialGraph,2);
Nodes = cell2mat(InitialGraph(3:4,2:Length));
FindMaxNode = max(Nodes(:));
%We find and clear unused nodes in a way so names of nodes have the right
%order (0,1,2,3,...No_Of_Nodes) and make new graph with correct node names
FirstUnused = FindUnusedNodes(Nodes);
if ~(isempty(FirstUnused))
    [Changes,Nodes]= ClearUnused(Nodes,FirstUnused);
end
MyGraph = MakeMyGraph(Nodes, InitialGraph, Length)

size1 = 0;
if size(DesignVector,2)>1
    Ids = cell2mat(MyGraph(5,2:Length));
    [MyGraph(2,2:Length),MyDesignVector]=
GiveDesignVals(MyGraph(2,2:Length),DesignVector,Ids);
    size1 =  size(MyDesignVector(1,2:size(MyDesignVector,2)),2);
else
    MyDesignVector = DesignVector;
end

%We make the I and V Graphs of the circuit using the modified nodal
%equations using two graphs
[IGraph,VGraph,IMap,VMap] = MakeI_VGraphs( MyGraph ,Nodes );
ILength = size(IGraph,2);
INodes = cell2mat(IGraph(3:4,2:ILength));
FirstUnusedI = FindUnusedNodes(INodes);
if ~(isempty(FirstUnusedI))
    [ChangesI,INodes]= ClearUnused(INodes,FirstUnusedI);
    IGraph = MakeMyGraph(INodes, IGraph, ILength);
end

VLength = size(VGraph,2);
VNodes = cell2mat(VGraph(3:4,2:VLength));
FirstUnusedV = FindUnusedNodes(VNodes);
if ~(isempty(FirstUnusedV))
    [ChangesV,VNodes]= ClearUnused(VNodes,FirstUnusedV);
    VGraph = MakeMyGraph(VNodes, VGraph, VLength);
end
%--------I and V Graphs constructed-------------%
[SensitivityVector, W, WhoFi, WhoZ, D, Adm,Cap] =
MakeTea(MyGraph,IMap,VMap,Nodes);

size2 = 0;
if ~isempty(SensitivityVector)
    size2 = size(SensitivityVector,2);
end
assert(size1==size2,'SensitivityVector and MyDesignVector size mismatch')

NFreqs = size(Frequencies,2);
z = zeros(size(WhoZ,1),1);
z_Adj = zeros(size(WhoZ,1),1);
Fi = zeros(NFreqs,size(WhoFi,1));
Sensitivities = zeros(NFreqs,(size(MyDesignVector,2)-1));
FrequencySensitivity = zeros(NFreqs,size(D,1));
```

```
w = sym('w');
T_freq = Adm + 1i*w*Cap

for k=1:NFreqs
    freq = Frequencies(k);
    T = Adm + 1i*freq*Cap;
    Tt = T.';
    z = T\W;
    z_Adj = Tt \ D.';
    Sensitivities(k,1:size(Sensitivities,2)) = CalcSens( z, z_Adj,
SensitivityVector, MyDesignVector, freq );
    FrequencySensitivity(k,1:size(D,1)) = (1i*(z_Adj.')*Cap*z).';
    Fi(k,1:size(Fi,2)) = D*z;
end


%-----------System solved and sensitivities found - > Now we show the
%results--------------------------------------------------------------

Frequencies = Frequencies.'; %to match the table's needs

%We will show the bode plots of each output divided by each input

SourceVector = FindSources(MyGraph);
[GainVector, GainNames, PhaseVector, PhaseNames] = MakeTF(SourceVector,Fi,
WhoFi);
BodeNames = {GainNames PhaseNames};
BodeNames = cell2mat(BodeNames);


%Here the table of sensitivities is constructed

%First we find Group Delay of the outputs

OutputNames = cell(1,size(WhoFi,1));
GroupDelayN = OutputNames;
for k = 1:size(WhoFi,1)
    OutputNames{1,k} = char(WhoFi(k));
    GroupDelayN{1,k}=strcat(char(WhoFi(k)),' GrpDelay');
end

%Then we match the table needs so it can be shown correctly as columns
NewBodeNames = {};
for k = 1:size(BodeNames,1)
    IND = size(BodeNames,2);
    if mode(IND,2)
        %IND is odd
        ind = round(IND/2);
    else
        %IND is even
        ind = IND/2+1;
    end
    new_cell = {BodeNames(k,1:ind) BodeNames(k,(ind+1):size(BodeNames,2))};
    NewBodeNames = [NewBodeNames new_cell];
end

Names = ['Frequencies' NewBodeNames OutputNames];
```

```
Values = [Frequencies GainVector PhaseVector Fi];
Table1 = [num2cell(Names); num2cell(Values)];
figure
cellplot(Table1)
title('Vout - Gain - Phase')
Names = ['Frequencies' GroupDelayN
MyDesignVector(1,2:size(MyDesignVector,2))];
Values = [Frequencies FrequencySensitivity Sensitivities];
Table2 = [num2cell(Names); num2cell(Values)];
figure
cellplot(Table2)
title('Sensitivities')

NBodePlots = size(NewBodeNames,2)/2;
figure
l=1;
for k = 1:2:(2*NBodePlots -1)
    assert(l<=NBodePlots,'Error in Bode Diagrams');
    subplot(NBodePlots,2,k)
    semilogx(Frequencies,GainVector(:,l));
    title(NewBodeNames(k))
    subplot(NBodePlots,2,k+1)
    semilogx(Frequencies,PhaseVector(:,l));
    title(NewBodeNames(k+1))
    l=l+1;
end

%-------------------------------------------------------------------------%
%InputPart takes input from the user all the passive elements from the
%desired circuit so it does all the calculations

function InputPart(~)

end

%-------------------------------------------------------------------------%
%GiveFreqVals asks the user to give the vector of frequencies in which the
%analysis will take place

function Frequencies = GiveFreqVals()
END = -1;
UserInp = -1;
Fidx = 1;
disp('Give frequency ranges\n');
StandardMsg1 = 'Input (Wmin,Wmax,Nsamples)\n';
StandardMsg2 = 'I have finished with input';
Frequencies = [];
while END == -1
    disp('You have two choices: \n');
    disp('1.');
    disp(StandardMsg1);
    disp('2.');
    disp(StandardMsg2);
    prompt = 'Your choice:';
    while (UserInp~=1) && (UserInp~=2)
        UserInp = input(prompt);
    end
```

```
    switch UserInp
        case 1
            Wmin = ParseVal('Wmin:');
            Wmax = ParseVal('Wmax:');
            N = ParseVal('Number of samples:');
            AddedFrequencies = linspace(Wmin,Wmax,N);
            Frequencies = [Frequencies AddedFrequencies];
            Fidx = Fidx + N;
        case 2
            END = 1;
    end
    UserInp = -1;
end
Frequencies = unique(Frequencies);
end

function Out = ParseVal(InpMsg)
    Out = -1;
    while Out<0 || ~(isa(Out,'double')) || isempty(Out)
        Out = input(InpMsg);
        if ~(isa(Out,'double'))
            disp('Only doubles please');
        elseif Out<0
            disp('Only positive values for this please');
        end
     end
end

%--------------------------------------------------------------------------%
%FindUnusedNodes finds which of the nodes hasn't been used by the user. The
%names of the nodes that user inputs must follow the sequence of the
%Natural numbers.

function FirstUnused = FindUnusedNodes( Nodes )
FindMaxNode = max(Nodes(:));
k = 0;
FirstUnused= -1;
while k<=FindMaxNode && FirstUnused==-1 %determine whether the element k
exist in Nodes
    if any(Nodes(:) == k)
        k = k+1;
    else
        FirstUnused = k;
    end
end
if FirstUnused == -1
    FirstUnused =[];
end
end

%--------------------------------------------------------------------------%
%[UnusedMap,MyNodes] = ClearUnused(Nodes,FirstUnused)
%This function takes a matrix of integers(Nodes) and return the same matrix
%but with some elements changed in a way that if the output matrix(MyNodes)
%had it's elements lie in a single row and sorted out with no repetitions
%then every element would satisfy the condition
%(SortedUniqueRow(i) == SortedUniqueRow(i+1) - 1)
```

```
%Nodes -> The input matrix
%FirstUnused -> The smallest integer that Nodes needs so it could be the
%desired matrix
%UnusedMap -> The mapping of changes that were done
%MyNodes   -> The desired matrix


function [UnusedMap,MyNodes] = ClearUnused( Nodes,FirstUnused )

MyNodes = Nodes;

NumToChange = FirstUnused;
NumsToBeChanged = (MyNodes(MyNodes>NumToChange));%All the elements in input
matrix that are larger than FirstUnused
NumsToBeChanged = unique(NumsToBeChanged);       %are sorted out in a row
(NumsToBeChanged) with no repetitions
Length = length(NumsToBeChanged);

UnusedMap(1:Length,1:2)=-1;
idx=1;

for k = 1:Length
    MyNodes(MyNodes==NumsToBeChanged(k))=NumToChange; %Every element of the
intersection of input matrix and NumsToBeChanged
    UnusedMap(idx,1:2) = [NumsToBeChanged(k) NumToChange]; % and the history
of changes are kept in UnusedMap
    idx=idx+1;
    NumToChange=NumToChange+1;
end
ClearMinus = UnusedMap(:,1)==-1;
UnusedMap(ClearMinus,:) =[];
end


%-------------------------------------------------------------------------%
%Finds the max number that the names of nodes have
function MaxNode = FindMaxNode(Cell_array)
Matrix = cell2mat(Cell_array);
MaxNode = max(Matrix(:));
end

%-------------------------------------------------------------------------%

function MyGraph = MakeMyGraph( InputNodes, Init, InitLength )
MyGraph = Init;
MyGraph(3:4,2:InitLength) = num2cell(InputNodes);
end

%-------------------------------------------------------------------------%
%Give Design Vals asks the user to give the desired values that the deisgn
%elements will have

function [ NewVals, MyDesVec ] = GiveDesignVals( MyInput, DesignVector,Ids )
NewVals = MyInput;
MyDesVec = DesignVector;
Inp = '';
StandardMsg = 'Give Input for :\n';
```

```
for k = 2:size(DesignVector,2);
    Name = DesignVector{1,k};
    ID = DesignVector{2,k};
    prompt = strcat(StandardMsg,Name);
    prompt = strcat(prompt,'\n');
    while isempty(Inp) || ~(isa(Inp,'double'))
        Inp = input(prompt);
        if ~(isa(Inp,'double'))
            disp('Only doubles please');
        else
            assert(isa(Inp,'double'),'I would like only doubles')
            CorrectCol = Ids==ID;
            NewVals{CorrectCol} = Inp;
            MyDesVec{3,k} = Inp;
        end
    end
    Inp = '';
end
end

%-------------------------------------------------------------------------%
%[IGraph,VGraph,IChanges,VChanges] = MakeI_VGraphs( Graph , Nodes)
%This function takes a matrix of integers(Nodes) and a cell array of cells
%(Graph) and gives two matrices (IGraph,VGraph) that corresponds to the I
%and V Graph of the circuit. It also outputs 2 maps to let us know where
%the node of Graph is currently in I and V graphs respectively
%Nodes - > 2xNo_Of_Edges matrix
%Graph - > 4x(No_Of_Edges+1) cell array, graph of circuit
%IGraph- > 4x(No_Of_Edges_In_I_Graph+1) cell array
%VGraph- > 4x(No_Of_Edges_In_V_Graph+1) cell array
%IMap - > (No_Of_Edges_In_I_Graph+1)x2 Mapping of nodes of Graph to IGraph
%VMap - > (No_Of_Edges_In_I_Graph+1)x2 Mapping of nodes of Graph in VGraph
function [IGraph,VGraph,IMap,VMap] = MakeI_VGraphs( Graph , Nodes)

NodeLength = size(Nodes,2);
GraphLength = size(Graph,2);

DeleteICols(1,1:NodeLength) = -1;
DeleteVCols(1,1:NodeLength) = -1;

MaxNode = max(max(Nodes));
InOrder = 0:MaxNode;
IMap = cell(MaxNode+1,2);
VMap = cell(MaxNode+1,2);

InOrdCell = num2cell(InOrder);
IMap(:,1) = InOrdCell;
IMap(:,2) = InOrdCell;
VMap(:,1) = InOrdCell;
VMap(:,2) = InOrdCell;

Types = Graph(1,2:GraphLength);
IGraphNodes = Nodes;
VGraphNodes = Nodes;

idxI = 1;
idxV = 1;
```

```
for k = 1:NodeLength
   switch Types{k}
       case {'J', 'aOut','gOut'}
           DeleteVCols(idxV) = k+1;
           idxV = idxV +1;
       case {'V','mOut','rOut'}
           ICollapsed= max(IGraphNodes(1,k),IGraphNodes(2,k));
           ICollapsedTo = min(IGraphNodes(1,k),IGraphNodes(2,k));
           B = IGraphNodes == ICollapsed;
           IGraphNodes(B) = ICollapsedTo;
           assert(size(IMap{ICollapsedTo+1,1},2)==1)
           IMap{ICollapsed+1,1}=cat(2, ICollapsed, ICollapsedTo);
           if ~(isempty(IMap{ICollapsed+1,2}))
               IMap{ICollapsedTo+1,2} = cat(2, IMap{ICollapsedTo+1,2},
IMap{ICollapsed+1,2} );
           end
       case {'OC', 'gIn','mIn'}
           DeleteICols(idxI) = k+1;
           idxI = idxI +1;
       case {'SC','aIn','rIn'}
           VCollapsed= max(VGraphNodes(1,k),VGraphNodes(2,k));
           VCollapsedTo = min(VGraphNodes(1,k),VGraphNodes(2,k));
           B = VGraphNodes == VCollapsed;
           VGraphNodes(B) = VCollapsedTo;
           assert(size(VMap{VCollapsedTo+1,1},2)==1)
           VMap{VCollapsed+1,1}=cat(2, VCollapsed, VCollapsedTo);
           if ~(isempty(VMap{VCollapsed+1,2}))
                VMap{VCollapsedTo+1,2} = cat(2, VMap{VCollapsedTo+1,2},
VMap{VCollapsed+1,2} );
           end
       case {'Null', 'OpIn'}
           DeleteICols(idxI) = k+1;
           idxI = idxI +1;
           VCollapsed= max(VGraphNodes(1,k),VGraphNodes(2,k));
           VCollapsedTo = min(VGraphNodes(1,k),VGraphNodes(2,k));
           B = VGraphNodes == VCollapsed;
           VGraphNodes(B) = VCollapsedTo;
           assert(size(VMap{VCollapsedTo+1,1},2)==1)
           VMap{VCollapsed+1,1}=cat(2, VCollapsed, VCollapsedTo);
           if ~(isempty(VMap{VCollapsed+1,2}))
               VMap{VCollapsedTo+1,2} = cat(2, VMap{VCollapsedTo+1,2},
VMap{VCollapsed+1,2} );
           end
       case {'Nora', 'OpOut'}
           DeleteVCols(idxV) = k+1;
           idxV = idxV +1;
           ICollapsed= max(IGraphNodes(1,k),IGraphNodes(2,k));
           ICollapsedTo = min(IGraphNodes(1,k),IGraphNodes(2,k));
           B = IGraphNodes == ICollapsed;
           IGraphNodes(B) = ICollapsedTo;
           assert(size(IMap{ICollapsedTo+1,1},2)==1)
           IMap{ICollapsed+1,1}=cat(2, ICollapsed, ICollapsedTo);
           if ~(isempty(IMap{ICollapsed+1,2}))
               IMap{ICollapsedTo+1,2} = cat(2, IMap{ICollapsedTo+1,2},
IMap{ICollapsed+1,2} );
           end
```

```
   end
end

IGraph = MakeMyGraph( IGraphNodes, Graph, GraphLength) ;
VGraph = MakeMyGraph( VGraphNodes, Graph, GraphLength) ;

%Now we have full I and V Graphs and we need to delete the columns asserted
%by DeleteICols (cause of OpenCircuits) either the ones that
%form Self loops (e.g. the branch that went from node 0 to 1 and was the
%only branch in the node now makes a self loop from 0 to 0 and is
%implicitly deleted from the I Graph)

%First we clear the rows DeleteICols and DeleteVCols from their -1's
ClearMinus = DeleteICols==-1;
DeleteICols(ClearMinus) =[];
ClearMinus = DeleteVCols==-1;
DeleteVCols(ClearMinus) =[];

%Now we input a logic of or's to 2 logical rows for either graphs

%For the I Graph we make a row of 0's and we fill with 1's the positions
%indicated by DeleteICols
indexI = false(1,GraphLength);
indexI(DeleteICols) = true;

%Afterwards we make a row that finds the columns in IGraph that form self
%loops in a sense that destination node is identical to arrival node
%This is also a logical row that we pad it with one 0 at the beginning so
%it has the correct dimension (dimension of previous logical row)
INewNodes = cell2mat(IGraph(3:4,2:size(IGraph,2)));
DeleteSelfILoops = INewNodes(1,:) == INewNodes(2,:);
DeleteSelfILoops = cat(2,0,DeleteSelfILoops);


%Implement or and have the inverse show us the columns that shouldn't exist
%in IGraph
DeleteI = or(indexI,DeleteSelfILoops);
DeleteI = ~DeleteI;

%If a change has to be made (The result of OR gave us some columns to be
%deleted) make IGraph the desired one
if ~(all(DeleteI == true))
    IGraph = IGraph(:,DeleteI);
end
%Do the same for V Graph

%Make a row of 0's and we fill with 1's the positions indicated by
%DeleteVCols
indexV = false(1,GraphLength);
indexV(DeleteVCols) = true;

%Make a row that finds the columns in VGraph that form self loops and make
%the correct padding again
VNewNodes = cell2mat(VGraph(3:4,2:size(VGraph,2)));
DeleteSelfVLoops = VNewNodes(1,:) == VNewNodes(2,:);
DeleteSelfVLoops = cat(2,0,DeleteSelfVLoops);
```

```
%Implement same or as before for VGraph
DeleteV = or(indexV,DeleteSelfVLoops);
DeleteV = ~DeleteV;
if ~(all(DeleteV == true))
    VGraph = VGraph(:,DeleteV);
end

%Clean IMaps and VMaps from unused nodes
index = true(1,size(IMap,1));
for k = 1:size(IMap,1)
    if size(IMap{k,1},2) > 1
        index(k) = false;
    end
end
IMap = IMap(index,:);

index = true(1,size(VMap,1));
for k = 1:size(VMap,1)
    if size(VMap{k,1},2) > 1
        index(k) = false;
    end
end
VMap = VMap(index,:);

FirstCol = 0:(size(IMap,1)-1);
IMap(:,1) = num2cell(FirstCol);
FirstCol = 0:(size(VMap,1)-1);
VMap(:,1) = num2cell(FirstCol);
end


%-------------------------------------------------------------------------%
%Makes the T matrix of the circuit which is equivalent in developping the
%equations that are essential for analyzing the circuit
function [SensitivityVector, W, Fi, z, D, Adm,Cap] =
MakeTea(Graph,IMap,VMap,Nodes)
ISize = size(IMap,1)-1;
VSize = size(VMap,1)-1;
Adm = zeros(ISize,VSize);
Cap = Adm;
W = zeros(ISize,1);
D = zeros(1,VSize);
z = sym('V',[VSize 1]);
Fi = sym(zeros(1,1));

NodeLength = size(Nodes,2);
GraphLength = size(Graph,2);

Type = Graph(1,2:GraphLength);
Value = Graph(2,2:GraphLength);
Ids = cell2mat(Graph(5,2:GraphLength));
SensitivityVector = zeros(4,0);

Widx=ISize+1;
DRows = 1;
DCols = size(D,2) + 1;
```

```
Zidx = 1 + size(z,1);
TRows=ISize+1;
TCols=VSize+1;
Fidx=1;
Sidx=1;

k=1;
while k <=NodeLength
    Val =Value{k};
    switch Type{k}
        case 'G'
            [ji1,ji2] = FindPos(IMap,Nodes(1,k),Nodes(2,k));
            [jv1,jv2] = FindPos(VMap,Nodes(1,k),Nodes(2,k));

            if ji1 ~=0
                if jv1 ~=0
                    Adm(ji1,jv1) = Adm(ji1,jv1) + Val;
                end
                if jv2 ~=0
                    Adm(ji1,jv2) = Adm(ji1,jv2) - Val;
                end
            end
            if ji2 ~=0
                if jv1 ~=0
                    Adm(ji2,jv1) = Adm(ji2,jv1) - Val;
                end
                if jv2 ~=0
                    Adm(ji2,jv2) = Adm(ji2,jv2) + Val;
                end
            end
        case 'C'
            [ji1,ji2] = FindPos(IMap,Nodes(1,k),Nodes(2,k));
            [jv1,jv2] = FindPos(VMap,Nodes(1,k),Nodes(2,k));
            if ji1 ~=0
                if jv1 ~=0
                    Cap(ji1,jv1) = Cap(ji1,jv1) + Val;
                end
                if jv2 ~=0
                    Cap(ji1,jv2) = Cap(ji1,jv2) - Val;
                end
            end
            if ji2 ~=0
                if jv1 ~=0
                    Cap(ji2,jv1) = Cap(ji2,jv1) - Val;
                end
                if jv2 ~=0
                    Cap(ji2,jv2) = Cap(ji2,jv2) + Val;
                end
            end
        case 'gIn'
            [jv1,jv2] = FindPos(VMap,Nodes(1,k),Nodes(2,k));
            k=k+1;
            [ki1,ki2] = FindPos(IMap,Nodes(1,k),Nodes(2,k));
            Val =Value{k};
            if ki1 ~=0
                if jv1 ~=0
                    Adm(ki1,jv1) = Adm(ki1,jv1) + Val;
```

```
            end
            if jv2 ~=0
                Adm(ki1,jv2) = Adm(ki1,jv2) - Val;
            end
        end
        if ki2 ~=0
            if jv1 ~=0
                Adm(ki2,jv1) = Adm(ki2,jv1) - Val;
            end
            if jv2 ~=0
                Adm(ji2,jv2) = Adm(ji2,jv2) + Val;
            end
        end
    case 'J'
        [ji1,ji2] = FindPos(IMap,Nodes(1,k),Nodes(2,k));
        if ji1~=0
            W(ji1) = W(ji1) - Val;
        end
        if ji2~=0
            W(ji2) = W(ji2) + Val;
        end
        [jv1, jv2] = deal(0,0);
    case 'V'
        [jv1,jv2] = FindPos(VMap,Nodes(1,k),Nodes(2,k));
        W(Widx,1) = Val;
        Widx=Widx+1;
        NewHorVec = zeros(1,size(Adm,2));
        Cap(TRows,:) = NewHorVec;
        if jv1~=0
            NewHorVec(jv1) = 1;
        end
        if jv2~=0
            NewHorVec(jv2) = -1;
        end
        Adm(TRows,:) = NewHorVec;
        [ji1, ji2] = deal(TRows,0);
        TRows=TRows+1;
    case 'Z'
        W(Widx,1) = Val;
        Widx=Widx+1;

        NewD = zeros(size(D,1),1);
        D(:,DCols) = NewD;
        DCols = DCols +1;

        [ji1,ji2] = FindPos(IMap,Nodes(1,k),Nodes(2,k));
        [jv1,jv2] = FindPos(VMap,Nodes(1,k),Nodes(2,k));

        PrevSize = size(Adm,2);
        NewHorVec = zeros(1,PrevSize);
        Cap(TRows,:) = NewHorVec;
        if jv1~=0
            NewHorVec(jv1) = 1;
        end
        if jv2~=0
            NewHorVec(jv2) = -1;
        end
```

```
        Adm(TRows,:) = NewHorVec;
        TRows = TRows+1;

        NewSize = size(Adm,1);
        AdmVerVec = zeros(NewSize,1);
        if ji1~=0
            AdmVerVec(ji1) = 1;
        end
        if ji2~=0
            AdmVerVec(ji2) = -1;
        end
        Adm(:,TCols) = AdmVerVec;

        CapVerVec = zeros(NewSize,1);
        [ji1,ji2,jv1,jv2] = deal(NewSize,0,TCols,0);
        CapVerVec(NewSize) = -Val;
        Cap(:,TCols) = CapVerVec;
        TCols = TCols + 1;
    case 'mIn'
        W(Widx,1) = 0;
        Widx=Widx+1;

        [jv1,jv2] = FindPos(VMap,Nodes(1,k),Nodes(2,k));
        k=k+1;
        [kv1,kv2] = FindPos(VMap,Nodes(1,k),Nodes(2,k));
        Val =Value{k};
        NewHorVec = zeros(1,size(Adm,2));
        Cap(TRows,:) = NewHorVec;
        if jv1~=0
            NewHorVec(jv1) = -Val;
        end
        if jv2~=0
            NewHorVec(jv2) = Val;
        end
        if kv1~=0
            NewHorVec(kv1) = 1;
        end
        if kv2~=0
            NewHorVec(kv2) = -1;
        end
        Adm(TRows,:) = NewHorVec;
        [ji1,ji2] = deal(TRows,0);
        TRows = TRows +1;
    case 'aIn'
        NewD = zeros(size(D,1),1);
        D(:,DCols) = NewD;
        DCols = DCols +1;

        [ji1,ji2] = FindPos(IMap,Nodes(1,k),Nodes(2,k));
        j1 = num2str(Nodes(1,k));
        j2 = num2str(Nodes(2,k));
        k=k+1;
        [ki1,ki2] = FindPos(VMap,Nodes(1,k),Nodes(2,k));
        Val =Value{k};
        NewVerVec = zeros(size(Adm,1),1);
        Cap(:,TCols) = NewVerVec;
        if ji1~=0
```

```
            NewVerVec(ji1) = 1;
        end
        if ji2~=0
            NewVerVec(ji2) = -1;
        end
        if ki1~=0
            NewVerVec(ki1) = Val;
        end
        if ki2~=0
            NewVerVec(ki2) = -Val;
        end
        Adm(:,TCols) = NewVerVec;
        [ji1,ji2,jv1,jv2] = deal(ki1,ki2,TCols,0);
        TCols = TCols + 1;

        Name = strcat('I',j1,'_',j2);
        z(Zidx,1) = sym(Name);
        Zidx = Zidx+1;
    case 'rIn'
        W(Widx,1) = Val;
        Widx=Widx+1;

        NewD = zeros(size(D,1),1);
        D(:,DCols) = NewD;
        DCols = DCols +1;

        [ji1,ji2] = FindPos(IMap,Nodes(1,k),Nodes(2,k));
        j1 = num2str(Nodes(1,k));
        j2 = num2str(Nodes(2,k));
        k=k+1;
        [kv1,kv2] = FindPos(VMap,Nodes(1,k),Nodes(2,k));
        Val =Value{k};
        NewHorVec = zeros(1,size(Adm,2));
        Cap(TRows,:) = NewHorVec;
        if kv1~=0
            NewHorVec(kv1) = 1;
        end
        if kv2~=0
            NewHorVec(kv2) = -1;
        end


        NewVerVec = zeros(size(Adm,1)+1,1);
        Cap(:,TCols) = NewVerVec;
        if ji1~=0
            NewVerVec(ji1) = 1;
        end
        if ji2~=0
            NewVerVec(ji2) = -1;
        end
        NewVerVec(size(Adm,1)+1) = -Val;
        Adm(:,TCols) = NewVerVec;
        Adm(TRows,:) = NewHorVec;

        TRows = TRows+1;
        TCols = TCols + 1;
        [ji1,ji2,jv1,jv2] = deal(TRows,0,TCols,0);
```

```
            Name = strcat('I',j1,'_',j2);
            z(Zidx,1) = sym(Name);
            Zidx = Zidx+1;
        case 'SC'
            [ji1,ji2] = FindPos(IMap,Nodes(1,k),Nodes(2,k));

            NewVerVec = zeros(size(Adm,1),1);
            Cap(:,TCols) = NewVerVec;
            if ji1~=0
                NewVerVec(ji1) = 1;
            end
            if ji2~=0
                NewVerVec(ji2) = -1;
            end
            Adm(:,TCols) = NewVerVec;
            TCols = TCols + 1;

            if Val == '-'
                j1 = num2str(Zidx);
                Name = strcat('I',j1);
                z(Zidx,1) = sym(Name);
                NewD = zeros(size(D,1),1);
                D(:,DCols) = NewD;
                DCols = DCols +1;
                Zidx = Zidx + 1;
            else
                assert(ischar(Val),'Error : Name of Output must be char' )
                z(Zidx,1) = sym(Val);
                Zidx = Zidx + 1;
                Fi(Fidx)= sym(Val);
                Fidx = Fidx + 1;
                D(DRows,:) = zeros(1,size(D,2));
                DRows = DRows + 1;
                NewD = zeros(size(D,1),1);
                NewD(size(D,1)) = 1;
                D(:,DCols) = NewD;
                DCols = DCols + 1;
            end
         case 'OC'
            [jv1,jv2] = FindPos(VMap,Nodes(1,k),Nodes(2,k));
            if Val ~= '-'
                assert(ischar(Val),'Error : Name of Output must be char' )
                Fi(Fidx) = sym(Val);
                Fidx = Fidx +1;

                D(DRows,:) = zeros(1,size(D,2));
                if jv1~=0
                    D(DRows,jv1) = 1;
                end
                if jv2~=0
                    D(DRows,jv2) = -1;
                end
                DRows = DRows +1;
            end
    end
    if Ids(k)~=0
```

```
        SensitivityVector(:,Sidx) = zeros(4,1);
        if ji1~=0
            SensitivityVector(1,Sidx) = ji1;
        end
        if ji2~=0
            SensitivityVector(2,Sidx) = ji2;
        end
        if jv1~=0
            SensitivityVector(3,Sidx) = jv1;
        end
        if jv2~=0
            SensitivityVector(4,Sidx) = jv2;
        end
        Sidx = Sidx+1;
    end
    k=k+1;
end
Fi = Fi.';
end

function [Out1, Out2] = FindPos(Map , K1, K2)
idx1 = cellfun(@(x) nnz(x==K1) > 0 ,Map(:,2));
idx2 = cellfun(@(x) nnz(x==K2) > 0 ,Map(:,2));
idx1 = idx1(:) == true;
idx2 = idx2(:) == true;
Out1 = Map{idx1,1};
Out2 = Map{idx2,1};
end


%-------------------------------------------------------------------------%
%Calculates the sensitivities of the outputs with respect to the design
%elements
function Sensitivities = CalcSens( Z, Za, SensVec, DesignVec, freq )
Length = size(DesignVec,2)-1;
Types = DesignVec(4,2:(Length+1));
Sensitivities = zeros(1,Length);
for k = 1:Length
    Name = Types{k};
    idxI = SensVec(1,k);
    idxJ = SensVec(2,k);
    idxK = SensVec(3,k);
    idxL = SensVec(4,k);
    if idxI~=0
        Zi = Za(idxI);
    else
        Zi = 0;
    end
    if idxJ~=0
        Zj = Za(idxJ);
    else
        Zj = 0;
    end
    if idxK~=0
        Zk = Z(idxK);
    else
        Zk = 0;
```

```
    end
    if idxL~=0
        Zl = Z(idxL);
    else
        Zl = 0;
    end
    switch Name
        case {'V', 'J'}
            Sensitivities(k) = Zj - Zi;
        case {'C', 'L'}
            Sensitivities(k) = freq*1i*(Zi - Zj)*(Zk - Zl);
        otherwise
            Sensitivities(k) = (Zi - Zj)*(Zk - Zl);
    end

end
end

%-------------------------------------------------------------------------%
%Finds which are the sources of the circuit so it can calculate the
%transfer function of each output with respect to each input
function SourceV = FindSources( Graph )
SourceV = {'Names'; 'Values'};
Types = Graph(1,2:size(Graph,2));
Values= Graph(2,2:size(Graph,2));
for k = 1:size(Types,2)
    switch Types{k}
        case {'J' ,'V'}
            NewCell = {Types{k}; Values{k}};
            SourceV = [SourceV NewCell];
    end
end
assert(size(SourceV,2)>1,'SourceV Empty')
end

%-------------------------------------------------------------------------%
%Calculates the transfer functions of the circuit
function [GainV, GainN, PhaseV, PhaseN] = MakeTF( SourceV, Fis, WhoFis )
GainV = zeros(size(Fis,1),1);
PhaseV= zeros(size(Fis,1),1);
Gidx = 1;
Pidx = 1;
Bidx = 1;
GainN = char(0);
PhaseN = char(0);
for k = 2:size(SourceV,2)
    In = SourceV{2,k};
    for l = 1:size(Fis,2)
        Out=Fis(:,l);
        OutN = char(WhoFis(l));
        InN  = num2str(SourceV{2,k});
        GainN(Bidx,1:(size(OutN,2) + size(InN,2) + 8))
=strcat(OutN,'/',InN,'-> Gain');
        PhaseN(Bidx,1:(size(OutN,2) + size(InN,2) +
8))=strcat(OutN,'/',InN,'->Phase');
        Tf = Out/In;
        GainV(1:size(Fis,1),Gidx) = 20*log(abs(Tf));
```

```
        PhaseV(1:size(Fis,1),Pidx)= atan(-(imag(Tf)./real(Tf)));
        Gidx = Gidx +1;
        Pidx = Pidx +1;
        Bidx = Bidx +1;
    end

end
end

%-------------------------------------------------------------------------%
 %Below is the code for all the elements that can be put in a circuit
%according to the program Driplomatiki
function c( K1,K2,val )
global InitialGraph DesignVector DesIdx;
Type = 'C';
if ischar(val)
    NewDesEl = {val; DesIdx; 0; Type};
    DesignVector = [DesignVector NewDesEl];

    new_cell = {Type; val; K1; K2; DesIdx};
    DesIdx = DesIdx+1;
else
    new_cell = {Type; val; K1; K2; 0};
end
InitialGraph = [InitialGraph new_cell];
end

%
function cs(K1, K2, val)
global InitialGraph DesignVector DesIdx;
Type = 'J';
if ischar(val)
    NewDesEl = {val; DesIdx; 0; Type};
    DesignVector = [DesignVector NewDesEl];

    new_cell = {Type; val; K1; K2; DesIdx};
    DesIdx = DesIdx+1;
else
    new_cell = {Type; val; K1; K2; 0};
end
InitialGraph = [InitialGraph new_cell];
end

%
function e(K1, K2, val)
global InitialGraph DesignVector DesIdx;
Type = 'V';
if ischar(val)
    NewDesEl = {val; DesIdx; 0; Type};
    DesignVector = [DesignVector NewDesEl];

    new_cell = {Type; val; K1; K2; DesIdx};
    DesIdx = DesIdx+1;
else
    new_cell = {Type; val; K1; K2; 0};
end
InitialGraph = [InitialGraph new_cell];
```

```
end

%
function g( K1,K2,val )
global InitialGraph DesignVector DesIdx;
Type = 'G';
if ischar(val)
    NewDesEl = {val; DesIdx; 0; Type};
    DesignVector = [DesignVector NewDesEl];

    new_cell = {Type; val; K1; K2; DesIdx};
    DesIdx = DesIdx+1;
else
    new_cell = {Type; val; K1; K2; 0};
end

InitialGraph = [InitialGraph new_cell];
end

%
function jtoj(K1, K2, K3, K4, val)
global InitialGraph DesignVector DesIdx;
Type1 = 'aIn';
Type2 = 'aOut';
if ischar(val)
    NewDesEl = {val; DesIdx; 0; Type2};
    DesignVector = [DesignVector NewDesEl];

    new_cell1 = {Type1; '-'; K1; K2; 0};
    new_cell2 = {Type2; val; K3; K4; DesIdx};
    DesIdx = DesIdx+1;
else
    new_cell1 = {Type1; '-'; K1; K2; 0};
    new_cell2 = {Type2; val; K3; K4; 0};
end
InitialGraph = [InitialGraph new_cell1 new_cell2];
end

%
function jtov(K1, K2, K3, K4, val)
global InitialGraph DesignVector DesIdx;
Type1 = 'rIn';
Type2 = 'rOut';
if ischar(val)
    NewDesEl = {val; DesIdx; 0; Type2};
    DesignVector = [DesignVector NewDesEl];

    new_cell1 = {Type1; '-'; K1; K2; 0};
    new_cell2 = {Type2; val; K3; K4; DesIdx};
    DesIdx = DesIdx+1;
else
    new_cell1 = {Type1; '-'; K1; K2; 0};
    new_cell2 = {Type2; val; K3; K4; 0};
end
InitialGraph = [InitialGraph new_cell1 new_cell2];
end
```

```
%
function l(K1, K2, val)
global InitialGraph DesignVector DesIdx;
Type = 'Z';
if ischar(val)
    NewDesEl = {val; DesIdx; 0; Type};
    DesignVector = [DesignVector NewDesEl];

    new_cell = {Type; val; K1; K2; DesIdx};
    DesIdx = DesIdx+1;
else
    new_cell = {Type; val; K1; K2; 0};
end
InitialGraph = [InitialGraph new_cell];
end

%
function oc(K1, K2, val)
global InitialGraph;
Type = 'OC';
new_cell = {Type; val; K1; K2; 0};
InitialGraph = [InitialGraph new_cell];
end

%
function opamp(K1, K2, K3, K4)
global InitialGraph;
Type1 = 'OpIn';
Type2 = 'OpOut';
new_cell1 = {Type1; '+-'; K1; K2; 0};
new_cell2 = {Type2; 'O'; K3; K4; 0};
InitialGraph = [InitialGraph new_cell1 new_cell2];
end

%
function r(K1, K2, val)
global InitialGraph DesignVector DesIdx;
Type = 'Z';
if ischar(val)
    NewDesEl = {val; DesIdx; 0; Type};
    DesignVector = [DesignVector NewDesEl];

    new_cell = {Type; val; K1; K2; DesIdx};
    DesIdx = DesIdx+1;
else
    new_cell = {Type; val; K1; K2; 0};
end
InitialGraph = [InitialGraph new_cell];
end

%
function sc(K1, K2, val)
global InitialGraph;
Type = 'SC';
new_cell = {Type; val; K1; K2; 0};
InitialGraph = [InitialGraph new_cell];
end
```

```
%
function vtoj(K1, K2, K3, K4, val)
global InitialGraph DesignVector DesIdx;
Type1 = 'gIn';
Type2 = 'gOut';
if ischar(val)
    NewDesEl = {val; DesIdx; 0; Type2};
    DesignVector = [DesignVector NewDesEl];

    new_cell1 = {Type1; '-'; K1; K2; 0};
    new_cell2 = {Type2; val; K3; K4; DesIdx};
    DesIdx = DesIdx+1;
else
    new_cell1 = {Type1; '-'; K1; K2; 0};
    new_cell2 = {Type2; val; K3; K4; 0};
end
InitialGraph = [InitialGraph new_cell1 new_cell2];
end

%
function vtov(K1, K2, K3, K4, val)
global InitialGraph DesignVector DesIdx;
Type1 = 'mIn';
Type2 = 'mOut';
if ischar(val)
    NewDesEl = {val; DesIdx; 0; Type2};
    DesignVector = [DesignVector NewDesEl];

    new_cell1 = {Type1; '-'; K1; K2; 0};
    new_cell2 = {Type2; val; K3; K4; DesIdx};
    DesIdx = DesIdx+1;
else
    new_cell1 = {Type1; '-'; K1; K2; 0};
    new_cell2 = {Type2; val; K3; K4; 0};
end
InitialGraph = [InitialGraph new_cell1 new_cell2];
end

%
function z(K1, K2, val)
global InitialGraph DesignVector DesIdx;
Type = 'Z';
if ischar(val)
    NewDesEl = {val; DesIdx; 0; Type};
    DesignVector = [DesignVector NewDesEl];

    new_cell = {Type; val; K1; K2; DesIdx};
    DesIdx = DesIdx+1;
else
    new_cell = {Type; val; K1; K2; 0};
end
InitialGraph = [InitialGraph new_cell];
end
```

Βιβλιογραφία – Αναφορές

[1] Ν. Μαράτος, *Σχεδίαση Γραμμικών Κυκλωμάτων*, ΕΜΠ, Αθήνα, 1989

[2] Τ.Γ. Κουσιουρής, *Θεωρία Ανάλυσης Συστημάτων και Κυκλωμάτων*, ΕΜΠ, Αθήνα

[3] Jiri Vlach & Kishore Singhal, *Computer Methods for Circuit Analysis and Design*, Van Nostrand Reinhold Company, Ontario, 1983