ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

# Αλγόριθμοι Σχεδίασης Ψευδοφάσματος Τετραγωνικών Πινάκων με χρήση Matlab και Python

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

**ΜΑΡΙΑΣ ΣΥΝΕΛΗ**

**Επιβλέπων:** Παναγιώτης Ψαρράκος, Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2023

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

# Αλγόριθμοι Σχεδίασης Ψευδοφάσματος Τετραγωνικών Πινάκων με χρήση Matlab και Python

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

**ΜΑΡΙΑΣ ΣΥΝΕΛΗ**

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή:

| Βασίλειος Κανελλόπουλος | Πέτρος Στεφανέας | Παναγιώτης Ψαρράκος |
|---|---|---|
| Καθηγητής Ε.Μ.Π. | Αναπλ. Καθηγητής Ε.Μ.Π. | Καθηγητής Ε.Μ.Π. (επιβλέπων) |

Αθήνα, Ιούνιος 2023

.....

**ΜΑΡΙΑ ΣΥΝΕΛΗ**

Διπλωματούχος Σχολής Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών Ε.Μ.Π.

# Περίληψη

Στη συγκεκριμένη διπλωματική εργασία θα ασχοληθούμε με αλγορίθμους εύρεσης του ψευδοφάσματος τετραγωνικών πινάκων. Το ψευδοφάσμα ενός πίνακα είναι μια κρίσιμη έννοια στην αριθμητική γραμμική άλγεβρα και σε συναφή πεδία όπως η θεωρία ελέγχου, η κβαντομηχανική, η δυναμική των ρευστών και άλλοι τομείς της επιστήμης των υπολογιστών. Το ψευδοφάσμα παρέχει πιο λεπτομερείς πληροφορίες από το φάσμα, διευκολύνοντας την πληρέστερη κατανόηση της συμπεριφοράς ενός τελεστή ή μιας δεδομένης μήτρας. Για να αποκτήσουμε μια διαισθητική κατανόηση του ψευδοφάσματος, χρησιμοποιούμε γραφικές μεθόδους υπολογισμού.

Το ψευδοφάσμα ενός πίνακα μπορεί να υπολογιστεί γραφικά και να οπτικοποιηθεί χρησιμοποιώντας μια ποικιλία μεθόδων και αλγορίθμων. Τρεις τέτοιοι αλγόριθμοι, ο αλγόριθμος GRID, ο αλγόριθμος Inclusion-Exclusion και ο αλγόριθμος Path-Following, χρησιμοποιούνται εκτενώς για το σκοπό αυτό.

Αυτή η εργασία παρουσιάζει μια υλοποίηση των αλγορίθμων GRID, Inclusion-Exclusion και Path-Following σε Matlab και Python. Οι GRID και Inclusion-Exclusion είναι ικανοί να αξιοποιούν πολλαπλούς επεξεργαστές, καθώς οι υπολογισμοί σε κάθε σημείο του πλέγματος είναι ανεξάρτητοι. Ο αλγόριθμος Path-Following ξεκινά με τον εντοπισμό ενός σημείου στην καμπύλη και στη συνέχεια επαναλαμβάνει την πρόβλεψη και τη διόρθωση του επόμενου σημείου.

Για τους αλγόριθμους GRID και Inclusion-Exclusion υπολογίζουμε τα ψευδοφάσματα σε πίνακες τυχαίων αριθμών, πίνακες Kahan και πίνακες Toeplitz GrCar. Παρέχουμε ακριβή χρονομέτρηση των αλγορίθμων και εξετάζουμε τις παραμέτρους που απαιτούνται για την εύρεση ικανής ακρίβειας.



**ΛέξειςΚλειδιά:** ψευδοφάσμα, grid, Inclusion-Exclusion, path-following

## Abstract

In this particular thesis we will deal with algorithms for finding the pseudospectrum of square matrices. The pseudospectrum of a matrix is a crucial concept in numerical linear algebra and related fields such as control theory, quantum mechanics, fluid dynamics, and other areas of computer science. The pseudospectrum provides more detailed information than the spectrum, facilitating a more complete understanding of the behavior of a given operator or matrix. To gain an intuitive understanding of the pseudospectrum, we use graphical calculation methods.

The pseudospectrum of a matrix can be graphically calculated and visualized using a variety of methods and algorithms. Three such algorithms, the GRID algorithm, the Inclusion-Exclusion algorithm, and the Path-Following algorithm, are extensively used for this purpose.

This thesis presents an implementation of the GRID, Inclusion-Exclusion, and Path-Following algorithms in Matlab and Python. GRID and Inclusion-Exclusion are capable of leveraging multiple processors, as computations at each grid point are independent. The Path-Following algorithm starts by locating a point on the curve and then repeats the prediction and correction of the next point.

For the GRID and Inclusion-Exclusion algorithms we compute pseudospectra in random number matrices, Kahan matrices and Toeplitz GrCar matrices. We provide precise timing of the algorithms and examine the parameters required to find sufficient accuracy.



Keywords: pseudospectrum, grid, Inclusion-Exclusion, path-following

## Ευχαριστίες

Με την ολοκλήρωση της διπλωματικής μου εργασίας αισθάνομαι την ιδιαίτερη υποχρέωση να εκφράσω τις θερμότερες ευχαριστίες μου στον επιβλέποντα και καθηγητή του Ε.Μ.Π. κ. Παναγιώτη Ψαρράκο, για την επίβλεψη της παρούσας διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να την εκπονήσω.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου που με στήριξαν σε πολλά επίπεδα κατά τη φοίτησή μου στο Εθνικό Μετσόβιο Πολυτεχνείο, όπως και όλους τους καθηγητές της Σ.Ε.Μ.Φ.Ε. που με καθοδήγησαν και με εισήγαγαν στον ευρύ και ξεχωριστό κόσμο των επιστημών της Φυσικής και των Μαθηματικών.

# Περιεχόμενα

# Εισαγωγή

Το ψευδοφάσμα ενός πίνακα είναι μια έννοια σημαντική στην αριθμητική γραμμική άλγεβρα και σε συναφή πεδία, όπως η θεωρία ελέγχου, η κβαντομηχανική, η δυναμική των ρευστών και άλλοι επιστημονικοί τομείς υπολογιστών. Ουσιαστικά, το ψευδοφάσμα παρέχει πιο λεπτομερείς πληροφορίες από το φάσμα μόνο, επιτρέποντας μια πλουσιότερη κατανόηση της συμπεριφοράς του τελεστή ή του εν λόγω πίνακα. Για να αποκτηθεί μια διαισθητική κατανόηση του ψευδοφάσματος, χρησιμοποιούνται γραφικές μέθοδοι υπολογισμού.

Οι ιδιοτιμές παρέχουν κρίσιμες πληροφορίες για τις ιδιότητες ενός πίνακα. Ωστόσο, μπορεί να είναι κάπως ανεπαρκείς στην παροχή ενός πλήρους χαρακτηρισμού του πίνακα, ιδιαίτερα στο πλαίσιο μη κανονικών πινάκων. Το ψευδοφάσμα, από την άλλη πλευρά, δίνει μια πιο λεπτομερή περιγραφή της συμπεριφοράς ενός πίνακα. Μαθηματικά, για έναν $n \times n$ πίνακα $A$ και έναν μη αρνητικό πραγματικό αριθμό ε, το ε-ψευδοφάσμα του $A$ είναι το σύνολο των μιγαδικών αριθμών z για τους οποίους $(zI - A)$ έχει μια μοναδική τιμή μικρότερη από ε. Με άλλα λόγια, το ψευδοφάσμα αντιπροσωπεύει μια περιοχή στο μιγαδικό επίπεδο όπου η νόρμα διαχωρισμού (το αντίστροφο της απόστασης από έναν μιγαδικό αριθμό στο φάσμα) είναι μεγάλη, παρέχοντας πληροφορίες για την ευαισθησία του προβλήματος της ιδιοτιμής.

Το ψευδοφάσμα ενός πίνακα μπορεί να υπολογιστεί γραφικά και να οπτικοποιηθεί χρησιμοποιώντας μια ποικιλία μεθόδων και αλγορίθμων. Τρεις τέτοιοι αλγόριθμοι, δηλαδή ο αλγόριθμος GRID, ο αλγόριθμος Inclusion-Exclusion και ο αλγόριθμος Path-Following, χρησιμοποιούνται εκτενώς για το σκοπό αυτό.

Στην παρούσα εργασία παραθέτουμε μια υλοποίηση του αλγόριθμου GRID σε Matlab και Python. Ο αλγόριθμος αυτός είναι ικανός να αξιοποιεί πολλαπλούς επεξεργαστές, καθώς οι υπολογισμοί σε κάθε σημείο του πλέγματος είναι ανεξάρτητοι. Στη συνέχεια, παραθέτουμε μια υλοποίηση του αλγόριθμου Inclusion-Exclusion σε Matlab και Python. Ο αλγόριθμος αυτός αποτελεί μια βελτίωση σε σχέση με τον αλγόριθμο GRID. Ο αλγόριθμος Inclusion-Exclusion ξεκινώντας με μια περιοχή που είναι γνωστό ότι περιέχει το ψευδοφάσμα, δημιουργεί «ζώνες αποκλεισμού» χωρίς σημεία ψευδοφασμάτων. Τέλος, παραθέτουμε μια υλοποίηση του αλγόριθμου Path-Following του Bruhl σε Matlab και Python. Ο αλγόριθμος Path-Following ξεκινά με τον εντοπισμό ενός σημείου στην καμπύλη και στη συνέχεια την πρόβλεψη και τη διόρθωση του επόμενου σημείου.

Το Κεφάλαιο 1,"Νόρμες Διανυσμάτων και Πινάκων", παρουσιάζει μια μικρή εισαγωγή στις βασικές μαθηματικές έννοιες που χρειάζονται στη συνέχεια της διπλωματικής εργασίας. Παρουσιάζονται κατά σειράν οι νόρμες διανυσμάτων, οι νόρμες πινάκων (συνήθεις και επαγόμενες) και η φασματική ακτίνα.

Το Κεφάλαιο 2,"SVD Παραγοντοποίηση Πίνακα", μας εισάγει στις παραγοντοποιήσεις πινάκων στη γραμμική άλγεβρα, χωρίζοντάς τις σε δύο κατηγορίες: αυτές που βοηθούν στην επίλυση συστημάτων γραμμικών εξισώσεων και αυτές που βασίζονται σε ιδιοτιμές. Στην SVD ένας πίνακας αναπαρίσταται ως $A = U D V^*$ με τον $D$ να είναι μη αρνητικός διαγώνιος πίνακας και οι $U$ και $V$ να ικανοποιούν συγκεκριμένες συνθήκες.

Το Κεφάλαιο 3, "Διαταραχές Ιδιοτιμών και Ψευδοφάσμα Πίνακα", εξετάζει τον ρόλο των ιδιοτιμών και του ψευδοφάσματος ενός πίνακα σε μαθηματικά και φυσικά συστήματα. Υπογραμμίζει τη σημασία της αποσύνθεσης τετραγωνικών πινάκων σε ιδιοτιμές και ιδιοδιανύσματα, μια διαδικασία που χρησιμοποιείται σε διάφορες εφαρμογές όπως η ανάλυση ευστάθειας και η ανάλυση δονήσεων περιστρεφόμενων σωμάτων. Ωστόσο, αναγνωρίζει ασυνέπειες με πειραματικές παρατηρήσεις όταν οι ιδιοτιμές παρουσιάζουν ανώμαλη συμπεριφορά σε σχέση με τον κανόνα του πίνακα από τον οποίο υπολογίστηκαν, σηματοδοτώντας μη κανονικούς πίνακες. Τέτοιοι πίνακες είναι ζωτικής σημασίας σε πολλά επιστημονικά πεδία. Η ανάγκη να μελετηθούν οι ιδιότητες των μη κανονικών πινάκων οδήγησε στον ορισμό του ψευδοφάσματος ενός πίνακα.

Το Κεφάλαιο 4 "Ο Αλγόριθμος GRID",παρουσιάζει μια υλοποίηση του αλγόριθμου GRID σε Matlab και Python. Ο GRID χρησιμοποιείται για την εύρεση του ψευδοφάσματος ενός πίνακα $A$. Αρχικά, δημιουργεί ένα πλέγμα (grid) μέσα σε ένα επιλεγμένο χωρίο του μιγαδικού επιπέδου, περιλαμβάνοντας σημεία z. Στη συνέχεια, υπολογίζει την $(zI - A)$ για κάθε z, καθορίζοντας αν το z ανήκει στο ψευδοφάσμα. Ο αλγόριθμος μπορεί να αξιοποιήσει πολλούς επεξεργαστές, καθώς οι υπολογισμοί σε κάθε σημείο του πλέγματος είναι ανεξάρτητοι. Εντούτοις, η επιτυχής εφαρμογή του αλγορίθμου εξαρτάται από την επιλογή του χωρίου Ω, το οποίο πρέπει να περιέχει το ψευδοφάσμα.

Το Κεφάλαιο 5 "Ο Αλγόριθμος Inclusion-Exclusion: Βελτίωση του Αλγορίθμου GRID",παρουσιάζει μια υλοποίηση του αλγόριθμου αυτούσε Matlab και Python.Ο αλγόριθμος Inclusion-Exclusion, που αναπτύχθηκε από τον Ιωάννη Κούτη και τον Ευστράτιο Γαλλόπουλο, είναι μια βελτίωση του αλγόριθμου GRID. Ο αλγόριθμος GRID είναι υπολογιστικά ακριβός λόγω του μεγάλου αριθμού. Ο αλγόριθμος Inclusion-Exclusion επιδιώκει να το ελαχιστοποιήσει αποκλείοντας ορισμένα σημεία από την εξέταση. Ξεκινώντας με μια περιοχή που είναι γνωστό ότι περιέχει το ψευδοφάσμα, χρησιμοποιεί τα αποτελέσματα SVD παραγοντοποιήσεων για να δημιουργήσει «περιοχές αποκλεισμού» χωρίς σημεία ψευδοφάσματος. Αυτός ο αλγόριθμος είναι λιγότερο χρονοβόρος, διατηρώντας παράλληλα την ακρίβεια της μεθόδου GRID.

Το Κεφάλαιο 6 "Συμπεράσματα και Μελλοντικές Προοπτικές",παρουσιάζει μια αποτίμηση για τους αλγόριθμους Grid και Inclusion-Exclusion και μια εισαγωγή στον αλγόριθμο Path-Following του Bruhl μαζί με μια υλοποίησή του αλγόριθμου αυτού σε Matlab και Python. Ο Path-Following εισήχθη από τον Kostin, η τεχνική εφαρμόστηκε για πρώτη φορά από τον M. Brühl, με μεθόδους παράλληλης ανίχνευσης καμπυλών που αναπτύχθηκαν αργότερα από τους Κ. Μπέκα και Ε. Γαλλόπουλο. Σε αντίθεση με τις προηγούμενες μεθόδους που χρησιμοποιούν ένα ομοιόμορφο τριγωνικό πλέγμα και βασίζονται σε τεχνικές διχοτόμησης, ο αλγόριθμος Path-Following ξεκινά με τον εντοπισμό ενός σημείου στην καμπύλη και στη συνέχεια προβλέπει και διορθώνει το επόμενο σημείο.

# 1. Νόρμες Διανυσμάτων και Πινάκων

## 1.1. Νόρμες Διανυσμάτων στο $\mathbb{C}^\nu$

**Ορισμός 1.1.1.** Μία συνάρτηση $\|\cdot\| : \mathbb{C}^\nu \to \mathbb{R}$ ονομάζεται *νόρμα διανυσμάτων* αν για κάθε $x, y \in \mathbb{C}^\nu$, ικανοποιεί τα ακόλουθα:

**(i)** $\|x\| \geq 0$ (μη αρνητική).

**(ii)** $\|x\| = 0$ αν και μόνο αν $x = 0$.

**(iii)** $\|a\,x\| = |a|\,\|x\|$ για κάθε $a \in \mathbb{C}$.

**(iv)** $\|x + y\| \leq \|x\| + \|y\|$ (τριγωνική ανισότητα).

**Ορισμός 1.1.2.** Μία νόρμα $\|\cdot\|$ ονομάζεται *ορθομοναδιαία αναλλοίωτη* αν για κάθε διάνυσμα $x \in \mathbb{C}^\nu$ και για κάθε ορθομοναδιαίο πίνακα $U \in \mathbb{C}^{\nu\times\nu}$ (δηλαδή, $U^*U = U\,U^* = I_\nu$), ισχύει $\|U\,x\| = \|x\|$.

**Ορισμός 1.1.3.** Μία συνάρτηση $\langle\cdot,\cdot\rangle : \mathbb{C}^\nu \times \mathbb{C}^\nu \to \mathbb{C}$ ονομάζεται *εσωτερικό γινόμενο* αν για κάθε $x, y, w \in \mathbb{C}^\nu$, ικανοποιεί τα ακόλουθα:

**(i)** $\langle x, x\rangle \geq 0$ (μη αρνητική).

**(ii)** $\langle x, x\rangle = 0$ αν και μόνο αν $x = 0$.

**(iii)** $\langle x + y, w\rangle = \langle x, w\rangle + \langle y, w\rangle$ (προσθετική).

**(iv)** $\langle a\,x, y\rangle = a\,\langle x, y\rangle$ για κάθε $a \in \mathbb{C}$.

**(v)** $\langle x, y\rangle = \overline{\langle y, x\rangle}$.

Από τον ορισμό του εσωτερικού γινομένου, εύκολα μπορεί κανείς να επιβεβαιώσει τις παρακάτω ιδιότητες:

**(1)** $\langle x, a\,y\rangle = \overline{\langle a\,y, x\rangle} = \overline{a}\,\overline{\langle y, x\rangle} = \overline{a}\,\langle x, y\rangle$.

**(2)** $\langle x, y + z\rangle = \overline{\langle y + z, x\rangle} = \overline{\langle y, x\rangle} + \overline{\langle z, x\rangle} = \langle x, y\rangle + \langle x, z\rangle$.

**(3)** $x = 0$ αν και μόνο αν $\langle x, y\rangle = 0$ για κάθε $y \in \mathbb{C}^\nu$.

**Πόρισμα 1.1.2.** *Για κάθε εσωτερικό γινόμενο $\langle\cdot,\cdot\rangle$ στο $\mathbb{C}^\nu$, η συνάρτηση $f(x) = \sqrt{\langle x, x\rangle}$ είναι νόρμα διανυσμάτων.*

## 1.2. Συνήθεις Νόρμες Διανυσμάτων στο $\mathbb{C}^\nu$

- Η $l_p$-*νόρμα* (ή $p$-*νόρμα*), για οποιονδήποτε πραγματικό αριθμό $p \geq 1$, ορίζεται ως

$$\|x\|_p = \left\|[x_1\ x_2\ \cdots\ x_\nu]^T\right\|_p = \left(|x_1|^p + |x_2|^p + \cdots + |x_\nu|^p\right)^{1/p}.$$

- Η *Ευκλείδια νόρμα* (ή $l_2$-*νόρμα*, ή 2-*νόρμα*) ορίζεται ως

$$\|x\|_2 = \left\|[x_1\ x_2\ \cdots\ x_\nu]^T\right\|_2 = \left(|x_1|^2 + |x_2|^2 + \cdots + |x_\nu|^2\right)^{1/2}.$$

- Η *αθροιστική νόρμα* (ή $l_1$-*νόρμα*, ή 1-*νόρμα*) ορίζεται ως

$$\|x\|_1 = \left\|[x_1\ x_2\ \cdots\ x_\nu]^T\right\|_1 = |x_1| + |x_2| + \cdots + |x_\nu|$$

και είναι γνωστή και ως *νόρμα του Μανχάταν*, καθώς συνδέεται με την απόσταση που διανύει ένα αυτοκίνητο κινούμενο πάνω στο ορθογώνιο πλέγμα δρόμων του Μανχάταν.

- Η *μεγιστική νόρμα* (ή max-*νόρμα*, ή $\infty$-*νόρμα*) ορίζεται ως

$$\|x\|_\infty = \left\|[x_1\ x_2\ \cdots\ x_\nu]^T\right\|_\infty = \max\{|x_1|, |x_2|, \ldots, |x_\nu|\}$$

## 1.3. Νόρμες Πινάκων στο $\mathbb{C}^{\nu x \nu}$

**Ορισμός 1.3.1.** Μία συνάρτηση $\|\circ\| : \mathbb{C}^{\nu\times\nu} \to \mathbb{R}$ ονομάζεται *νόρμα πινάκων* αν για κάθε $A, B \in \mathbb{C}^{\nu\times\nu}$, ικανοποιεί τα ακόλουθα:

**(i)** $\|A\| \geq 0$ (μη αρνητική).

**(ii)** $\|A\| = 0$ αν και μόνο αν $A = 0$.

**(iii)** $\|a\,A\| = |a|\,\|A\|$ για κάθε $a \in \mathbb{C}$.

**(iv)** $\|A + B\| \leq \|A\| + \|B\|$ (τριγωνική ανισότητα).

**(v)** $\|A\,B\| \leq \|A\|\,\|B\|$ (υπο-πολλαπλασιαστική ιδιότητα).

Ιδιότητες

Ειδικότερα, για κάθε νόρμα πινάκων $\|\circ\|$ και για το μοναδιαίο πίνακα $I_\nu$, ισχύει

$$\|I_\nu\| = \|I_\nu^2\| \leq \|I_\nu\|^2 \Rightarrow \|I_\nu\| \geq 1.$$

Επομένως, για κάθε αντιστρέψιμο πίνακα $A \in \mathbb{C}^{\nu\times\nu}$,

$$1 \leq \|I_\nu\| = \|A\,A^{-1}\| \leq \|A\|\,\|A^{-1}\| \Rightarrow \|A^{-1}\| \geq \frac{1}{\|A\|}.$$

## 1.4. Συνήθεις Νόρμες Πινάκων στο $\mathbb{C}^{\nu x \nu}$

- Η $l_1$-*νόρμα* ενός πίνακα $A = [a_{ij}] \in \mathbb{C}^{\nu\times\nu}$ ορίζεται ως

$$\|A\|_{l_1} = \sum_{i,j=1}^{\nu} |a_{ij}|.$$

- Η *νόρμα Frobenius* (ή $l_2$-*νόρμα*) ενός πίνακα $A = [a_{ij}] \in \mathbb{C}^{\nu\times\nu}$ ορίζεται ως

$$\|A\|_F = \left(\sum_{i,j=1}^{\nu} |a_{ij}|^2\right)^{1/2} = \sqrt{\mathrm{trace}(A^*A)},$$

όπου με $\mathrm{trace}(\cdot)$ συμβολίζουμε το ίχνος πίνακα.

- Η $l_\infty$-*νόρμα* ενός πίνακα $A = [a_{ij}] \in \mathbb{C}^{\nu\times\nu}$ ορίζεται ως

$$\|A\|_{l_\infty} = \max\{|a_{ij}| : i, j = 1, 2, \ldots, \nu\}.$$

## 1.5. Συνήθεις Επαγόμενες (Φυσικές) Νόρμες Πινάκων στο $\mathbb{C}^{\nu x \nu}$

Έστω $\|\cdot\|$ μία νόρμα διανυσμάτων στο $\mathbb{C}^\nu$. Η *επαγόμενη από την* $\|\cdot\|$ *νόρμα* στο $\mathbb{C}^{\nu\times\nu}$ ορίζεται ως

$$\|A\| = \max_{\|x\|=1} \|Ax\| = \max_{\|x\|\leq 1} \|Ax\| = \max_{x\neq 0} \frac{\|Ax\|}{\|x\|}.$$

- Η *νόρμα πινάκων μεγίστου αθροίσματος κατά στήλη* στο $\mathbb{C}^{\nu\times\nu}$ ορίζεται ως

$$\|A\|_1 = \max_{1\leq j\leq \nu} \sum_{i=1}^{\nu} |a_{ij}|.$$

Η νόρμα $\|\circ\|_1$ επάγεται από τη διανυσματική $l_1$-νόρμα. I

- Η *νόρμα πινάκων μεγίστου αθροίσματος κατά γραμμή* στο $\mathbb{C}^{\nu\times\nu}$ ορίζεται ως

$$\|A\|_\infty = \max_{1\leq i\leq \nu} \sum_{j=1}^{\nu} |a_{ij}|.$$

Η νόρμα $\|\circ\|_\infty$ επάγεται από τη διανυσματική $l_\infty$-νόρμα, δηλαδή

$$\|A\|_\infty = \max_{\|x\|_\infty=1} \|Ax\|_\infty.$$

- Η *φασματική (τελεστική) νόρμα πινάκων* στο $\mathbb{C}^{\nu\times\nu}$ ορίζεται ως

$$\|A\|_2 = \max\left\{\sqrt{\lambda} : \lambda \in \sigma(A^*A)\right\},$$

όπου με $\sigma(A)$ συμβολίζουμε το *φάσμα* ενός πίνακα $A \in \mathbb{C}^{\nu\times\nu}$, δηλαδή το σύνολο των ιδιοτιμών του $A$. Παρατηρούμε ότι αν $\lambda \in \sigma(A^*A)$ και $x \in \mathbb{C}^\nu$ ένα (μη μηδενικό) ιδιοδιάνυσμα του $A^*A$ που αντιστοιχεί στην ιδιοτιμή $\lambda$, τότε

$$(Ax)^*(Ax) = x^*A^*Ax = x^*\lambda x = \lambda(x^*x) \Rightarrow \|Ax\|_2^2 = \lambda\|x\|_2^2.$$

Επομένως, οι ιδιοτιμές του πίνακα $A^*A$ είναι μη αρνητικές κι έτσι μπορούν να οριστούν οι (μη αρνητικές) τετραγωνικές ρίζες τους. Επιπλέον, η φασματική νόρμα $\|\circ\|_2$ είναι νόρμα πινάκων η οποία επάγεται από την Ευκλείδια νόρμα $\|\cdot\|_2$, δηλαδή

$$\|A\|_2 = \max_{\|x\|_2=1} \|Ax\|_2,$$

και είναι ορθομοναδιαία αναλλοίωτη, δηλαδή $\|UAV\|_2 = \|A\|_2$ για κάθε ζεύγος ορθομοναδιαίων πινάκων $U, V \in \mathbb{C}^{\nu\times\nu}$.

## 1.6. Η Φασματική Ακτίνα

Έστω ένας πίνακας $A \in \mathbb{C}^{\nu\times\nu}$ με φάσμα $\sigma(A) = \{\lambda \in \mathbb{C} : \det(\lambda I_\nu - A) = 0\}$. Η *φασματική ακτίνα* του $A$ ορίζεται ως

$$\rho(A) = \max\{|\lambda| : \lambda \in \sigma(A)\}.$$

**Θεώρημα 1.4.1.** *Έστω $\|\circ\|$ μία νόρμα πινάκων στο $\mathbb{C}^{\nu\times\nu}$. Τότε για κάθε $A \in \mathbb{C}^{\nu\times\nu}$, ισχύει $\rho(A) \leq \|A\|$.*

Στο θεώρημα που ακολουθεί αποδεικνύεται ότι η φασματική ακτίνα $\rho(A)$ είναι το μέγιστο κάτω φράγμα (infimum) των νορμών πινάκων.

**Θεώρημα 1.4.3.** *Έστω ένας πίνακας $A \in \mathbb{C}^{\nu\times\nu}$ κι ένας πραγματικός αριθμός $\varepsilon > 0$. Τότε υπάρχει μία νόρμα πινάκων $\|\circ\|$ τέτοια ώστε $\|A\| \leq \rho(A) + \varepsilon$.*

# 2. SVD Παραγοντοποίηση Πίνακα

## 2.1. Γνωστές Παραγοντοποιήσεις

Στον μαθηματικό κλάδο της γραμμικής άλγεβρας, **παραγοντοποίηση (factorization) πίνακα ή αποσύνθεση (decomposition) πίνακα** είναι η έκφραση ενός πίνακα σε γινόμενο πινάκων. Υπάρχουν πολλές διαφορετικές παραγοντοποιήσεις πίνακα. Η καθεμιά βρίσκει χρήση σε μια συγκεκριμένη κατηγορία προβλημάτων. Υπάρχουν δύο βασικές κατηγορίες:

Παραγοντοποιήσεις που σχετίζονται με την επίλυση συστημάτων γραμμικών εξισώσεων:

- παραγοντοποίηση LU,
- παραγοντοποίηση κατάταξης,
- παραγοντοποίηση Cholesky,
- παραγοντοποίηση QR,
- παραγοντοποίηση RRQR,
- παραγοντοποίηση με παρεμβολή.

Παραγοντοποιήσεις βασισμένες σε ιδιοτιμές και σχετικές έννοιες:

- Ιδιοδιάσπαση, που ονομάζεται επίσης φασματική αποσύνθεση,
- παραγοντοποίηση Schur,
- παραγοντοποίηση QZ,
- παραγοντοποίηση Takagi,
- παραγοντοποίηση SVD,
- Αναλλοίωτες σε κλίμακα παραγοντοποιήσεις.

## 2.2. Παραγοντοποίηση SVD

Από τις δυνατές παραγοντοποιήσεις που υφίστανται για τους πίνακες, η σημαντικότερη, ως προς τις εφαρμογές, είναι η παραγοντοποίηση ιδιαζούσων τιμών(singular value decomposition - SVD).Εφαρμόζεται σε πίνακες m×n

$$A = U D V^*$$

όπου:

- ο *D* είναι ένας μη αρνητικός διαγώνιος πίνακας,
- οι πίνακε ς *U* και το *V* ικανοποιούν: $U^*U=I$, $V^*V=I$.
- Ο πίνακας $V^*$ είναι ο αναστροφοσυζυγής (conjugate transpose) του *V* (ή απλά ο ανάστροφος, αν ο *V* περιέχει μόνο πραγματικούς αριθμούς),
- ο *I* είναι ο ταυτοτικός πίνακας (κάποιας διάστασης).

Τα διαγώνια στοιχεία του *D* ονομάζονται ιδιάζουσες τιμές του πίνακα *A*.Οι ιδιάζουσες τιμές του *A* καθορίζονται πάντα μοναδικά. Οι πίνακες*U* και *V* δεν χρειάζεται να είναι μοναδικοί γενικά.

Αν *A* είναι ένας ν x μ μιγαδικός πίνακας, τότε υπάρχουν ορθομοναδιαίοι πίνακες $U \in \mathbb{C}^{\nu x \nu}$ και $V \in \mathbb{C}^{\mu x \mu}$ τέτοιοι ώστε

$$A = U \operatorname{diag}\{s_1, s_2, \ldots, s_{\min\{\nu,\mu\}}\} V^*$$

όπου$diag\{s_1, s_2, ..., s_{min\{\nu, \mu\}}\} \in \mathbb{C}^{\nu x \mu}$και$s_1 \geq s_2 \geq \ldots \geq s_{min\{\nu, \mu\}} \geq 0$.

Ο διαγώνιος πίνακας $diag\{s_1, s_2, ..., s_{min\{\nu, \mu\}}\}$ συμβολίζεται με Σ και οι τιμές

$s_1 \geq s_2 \geq \ldots \geq s_{min\{\nu, \mu\}} \geq 0$

καλούνται *ιδιάζουσες* τιμές του *A*. Αν γράψουμε $U = [u_1\ u_2\ \ldots\ u_\nu]$ και $V = [v_1\ v_2\ \ldots\ v_\mu]$, τότε τα διανύσματα στήλες $u_i$ και $v_i$ $(1 < i < min\{\nu, \mu\})$ λέγονται *αριστερά* και *δεξιά ιδιάζοντα διανύσματα* του *A* που αντιστοιχούν στην ιδιάζουσα τιμή $s_i$. Από τις σχέσεις $A\,V = U\,\Sigma$ και $A^*\,U = V\,\Sigma^T$ προκύπτει ότι

$$A\,v_i = s_i u_i \quad \text{και} \quad A^* u_i = s_i v_i, \quad \forall\, i = 1, 2, \ldots, \min\{\nu, \mu\}.$$

Ένα απλό παράδειγμα παραγοντοποίησης SVD είναι

$$A = \begin{bmatrix} 0.96 & 1.72 \\ 2.28 & 0.96 \end{bmatrix} = U \Sigma V^* = \begin{bmatrix} 0.6 & -0.8 \\ 0.8 & 0.6 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.8 & 0.6 \\ 0.6 & -0.8 \end{bmatrix}^*.$$

## 2.3. Ιδιάζουσες Τιμές και Δομή Πίνακα

Η παραγοντοποίηση SVD δίνει πολλές πληροφορίες για τη δομή ενός πίνακα. Αν θεωρήσουμε ότι η ιδιάζουσα τιμή sr είναι η ελάχιστη μη μηδενική ιδιάζουσα τιμή ενός $\nu \times \mu$ πίνακα $A$, δηλαδή αν

$s_1 \geq s_2, \ldots, \geq s_r \geq s_{r+1} = \ldots = s_{min\{\nu,\mu\}} = 0$

τότε $rank(A) = r$

Επίσης, $s_1 = s_1(A) = ||A||_2$

Επιπλέον, για $\nu > \mu$, ισχύει $\min_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = s_\mu$,

ενώ για $\nu = \mu$, $|\det(A)| = s_1 s_2 \cdots s_\nu$.

Οι μη μηδενικές ιδιάζουσες τιμές του $A$ είναι οι τετραγωνικές ρίζες των μη μηδενικών ιδιοτιμών των (θετικά ημιορισμένων) ερμιτιανών πινάκων $AA^*$ και $A^*A$.

Τα αριστερά ιδιάζοντα διανύσματα του $A$ είναι τα ιδιοδιανύσματα του $AA^*$.

Τα δεξιά ιδιάζοντα διανύσματα του $A$ είναι τα ιδιοδιανύσματα του $A^*A$.

# 3. Διαταραχές Ιδιοτιμών και Ψευδοφάσμα Πίνακα

## 3.1. Εισαγωγή

Οι ιδιοτιμές είναι ένα ειδικό σύνολο βαθμωτών τιμών που σχετίζονται με ένα γραμμικό σύστημα εξισώσεων. Η μελέτη των ιδιοτιμών τετραγωνικών πινάκων είναι ένα ιδιαίτερα σημαντική και αποτελεί το κυριότερο εργαλείο με το οποίο διαπιστώνονται οι ιδιότητες κάθε πίνακα ή τελεστή. Κάθε ιδιοτιμή συνδυάζεται με ένα αντίστοιχο λεγόμενο ιδιοδιάνυσμα. Ο προσδιορισμός των ιδιοτιμών και των ιδιοδιανυσμάτων ενός συστήματος είναι εξαιρετικά σημαντικός στη φυσική και τη μηχανική, όπου είναι ισοδύναμος με τη διαγωνοποίηση πίνακα και προκύπτει σε κοινές εφαρμογές όπως η ανάλυση ευστάθειας, η φυσική των περιστρεφόμενων σωμάτων και οι μικρές ταλαντώσεις των δονούμενων συστημάτων. Η αποσύνθεση ενός τετράγωνου πίνακα $A$ σε ιδιοτιμές και ιδιοδιανύσματα είναι γνωστή ως αποσύνθεση ιδιοτιμών.

Ωστόσο, η μελέτη των ιδιοτιμών ενδέχεται μη συμφωνεί με τις πειραματικές παρατηρήσεις. Αυτό συμβαίνει όταν οι ιδιοτιμές παρουσιάζουν ασυνήθιστη συμπεριφορά, σε σχέση με τη νόρμα πίνακα με βάση την οποία υπολογίστηκαν. Η νόρμα που χρησιμοποιείται συνήθως είναι η φασματική, η νόρμα δηλαδή που επάγεται από την Ευκλείδεια διανυσματική νόρμα (νόρμα-2). Στη συγκεκριμένη νόρμα, ασυνήθιστη συμπεριφορά σημαίνει ότι ο πίνακας είναι μη-κανονικός, δεν ισχύει δηλαδή η ισότητα $AA^* = A^*A$. Επίσης, τα ιδιοδιανύσματά του δεν είναι ορθογώνια.

Η μελέτη μη-κανονικών πινάκων είναι πολύ σημαντική και συναντάται σε πολλούς επιστημονικούς τομείς. Όσον αφορά τα Μαθηματικά, εφαρμόζεται στη Συναρτησιακή Ανάλυση, στην Αριθμητική Γραμμική Άλγεβρα, στις Στοχαστικές Ανελίξεις, στη θεωρία Μαρκοβιανών αλυσίδων και στις Διαφορικές Εξισώσεις. Εκτός των Μαθηματικών, εφαρμόζεται μεταξύ άλλων στην Ρευστομηχανική, την Υδροδυναμική, και τη Μετεωρολογία.

Η ανάγκη μελέτης των ιδιοτήτων των μη-κανονικών πινάκων οδήγησε στο ορισμό του ε-ψευδοφάσματος (ή απλώς ψευδοφάσματος) ενός πίνακα, μιας νέας έννοιας στη μελέτη των ιδιοτιμών [ΔΕΣΥΠΡΗ2018].

## 3.2. Ιδιοτιμές και Ιδιοδιανύσματα Πίνακα

Ενα **ιδιοδιάνυσμα** ενός τετραγωνικού πίνακα *A* είναι ένα μη μηδενικό διάνυσμα v που, όταν πολλαπλασιαστεί με τον *A*, ισούται με το αρχικό διάνυσμα, πολλαπλασιασμένο με έναν αριθμό $\lambda$ , έτσι ώστε:

$$Av = \lambda v$$

Ο αριθμός $\lambda$ ονομάζεται **ιδιοτιμή** του *A* που αντιστοιχεί στο *v*.

Στην αναλυτική γεωμετρία, για παράδειγμα, ένα διάνυσμα με 3 στοιχεία, μπορεί να ταυτιστεί με ένα βέλος σε ένα τρισδιάστατο χώρο, ξεκινώντας από την αρχή των αξόνων. Σ'αυτήν την περίπτωση, ένα ιδιοδιάνυσμα ενός 3x3 πίνακα *A* είναι ένα βέλος η κατεύθυνση του οποίου ή διατηρείται, ή γίνεται ακριβώς η αντίθετη, μετά τον πολλαπλασιασμό με τον *A*. Η αντίστοιχη ιδιοτιμή είναι αυτή που καθορίζει πως αλλάζει το μήκος του βέλους από τη διαδικασία, και εάν η κατεύθυνση του αντιστρέφεται ή όχι.

Στην αφηρημένη γραμμική άλγεβρα, οι έννοιες αυτές συνήθως επεκτείνονται σε πιο γενικές καταστάσεις, όπου οι παράγοντες που χρησιμοποιούνται σε πραγματική κλίμακα, αντικαθίστανται από σώματα κάθε διάστασης (όπως για παράδειγμα οι αλγεβρικοί ή οι μιγαδικοί αριθμοί), οι καρτεσιανές συντεταγμένες $R^n$ που αντικαθίστανται από τυχαίους διανυσματικούς χώρους (όπως για παράδειγμα των συνεχών συναρτήσεων, των πολυωνύμων ή των τριγωνομετρικών σειρών), και ο πολλαπλασιασμός πινάκων που αντικαθίσταται από κάθε γραμμικό τελεστή που απεικονίζει διανύσματα σε διανύσματα (όπως η παράγωγος από το διαφορικό λογισμό).

Το σύνολο όλων των ιδιοδιανυσμάτων ενός πίνακα (ή γραμμικού τελεστή), με το καθένα να ταιριάζει στην αντίστοιχη ιδιοτιμή του, καλείται το "ιδιοσύστημα" του πίνακα αυτού. Ο ιδιοχώρος ενός πίνακα *A* είναι το σύνολο όλων των ιδιοδιανυσμάτων με την ίδια ιδιοτιμή, συμπεριλαμβανομένου και του μηδενικού διανύσματος. Μια ιδιοβάση του *A* είναι κάθε βάση του συνόλου όλων των διανυσμάτων που αποτελείται απο γραμμικά ανεξάρτητα ιδιοδιανύσματα του *A*. Ενας πίνακας με στοιχεία πραγματικούς αριθμούς μπορεί να μην έχει καμία ιδιοτιμή, αλλά ένας πίνακας με στοιχεία μιγαδικούς αριθμούς, έχει πάντα τουλάχιστον μία μιγαδική ιδιοτιμή.

Οι ιδιοτιμές και τα ιδιοδιανύσματα έχουν πολλές εφαρμογές και στα θεωρητικά, αλλά και στα εφαρμοσμένα μαθηματικά. Χρησιμοποιούνται στην παραγοντοποίηση πινάκων, στην Κβαντική Μηχανική, και σε πολλούς άλλους τομείς.

Έστω $A$ ένας πίνακας $NxN$ με πραγματικούς ή μιγαδικούς συντελεστές. Έστω $\boldsymbol{v}$ ένα μη μηδενικό πραγματικό ή μιγαδικό διάνυσμα στήλης μήκους $N$, και έστω $\lambda$ πραγματική ή μιγαδική βαθμωτή τιμή. Τότε το $\boldsymbol{v}$ είναι ένα ιδιοδιάνυσμα του $A$ και το $\lambda$ είναι η αντίστοιχη ιδιοτιμή του, αν

$$A\,v \;=\; \lambda\,v.$$

Ακόμα κι αν το $A$ είναι πραγματικό, οι ιδιοτιμές του είναι γενικά μιγαδικές εκτός αν το $A$ είναι αυτοσυνημμένο (self-adjoint) Το σύνολο όλων των ιδιοτιμών του $A$ είναι το *φάσμα* του $A$, ένα μη κενό υποσύνολο του μιγαδικού επιπέδου που συμβολίζουμε με $\sigma(A)$. Το φάσμα μπορεί επίσης να οριστεί ως το σύνολο των σημείων $z \in E$ όπου ο επιλύων πίνακας (resolvent matrix),

$$(z - A)^{-1}$$

δεν υπάρχει. Το $z - A$ είναι συντομογραφία του $zI - A$, όπου $I$ είναι ο ταυτοτικός πίνακας.

Σε αντίθεση με τις μοναδικές τιμές (singular values), οι ιδιοτιμές έχουν νόημα μόνο για έναν πίνακα που είναι τετράγωνος. Αυτό αντικατοπτρίζει το γεγονός ότι σε εφαρμογές χρησιμοποιούνται όταν ένας πίνακας πρόκειται να συνδυαστεί επαναληπτικά, για παράδειγμα, ως δύναμη $A^k$ ή ως $e^{tA} = 1 + tA + 1/2(tA)^2 + ...$

Για τους περισσότερους πίνακες $A$, υπάρχει ένα *πλήρες σύνολο ιδιοδιανυσμάτων*, ένα σύνολο $N$ γραμμικά ανεξάρτητων διανυσμάτων $v_1, ..., v_N$ με $Av_j = \lambda_j v_j$. Εάν το $A$ έχει $N$ διακριτές ιδιοτιμές, τότε είναι εγγυημένο ότι έχει ένα πλήρες σύνολο από ιδιοδιανύσματα, που είναι μοναδικά μέχρι την κανονικοποίηση από βαθμωτούς παράγοντες. Για οποιονδήποτε πίνακα $A$ με πλήρες σύνολο ιδιοδιανυσμάτων $\{v_j\}$, έστω $V$ ο πίνακας $NxN$ του οποίου η $j$ στήλη είναι $V_j$, ένας *πίνακας ιδιοδιανυσμάτων*. Τότε μπορούμε να γράψουμε όλες τις $N$ συνθήκες ιδιοτιμών ταυτόχρονα με την εξίσωση πινάκων

$$AV = V\Lambda,$$

όπου $\boldsymbol{\Lambda}$ είναι ο διαγώνιος $NxN$ πίνακας του οποίου η $j_\eta$ διαγώνια καταχώρηση είναι $A_j$. Οπτικά:

$$\mathbf{A}\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_N \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_N \end{bmatrix}\begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{pmatrix}.$$

Δεδομένου ότι τα ιδιοδιανύσματα $Vj$ είναι γραμμικά ανεξάρτητα, ο $V$ είναι nonsingular, και έτσι μπορούμε να πολλαπλασιάσουμε το δεξί μέλος με $V^{-1}$ για να λάβουμε την παραγοντοποίηση

$$A = V \Lambda V^{-1}$$

γνωστή ως αποσύνθεση ιδιοτιμών ή διαγωνοποίηση του $A$. Λαμβάνοντας υπόψη αυτόν τον τύπο, ένας πίνακας με ένα πλήρες σύνολο ιδιοδιανυσμάτων λέγεται ότι μπορεί να διαγωνοποιηθεί (είναι διαγωνοποιήσιμος).

Η αποσύνθεση ιδιοτιμών εκφράζει μια αλλαγή βάσης σε «συντεταγμένες ιδιοδιανύσματος», δηλαδή συντελεστές σε μια επέκταση ιδιοδιανυσμάτων. Για παράδειγμα, αν $A=V \Lambda V^{-1}$, τότε έχουμε

$$V^{-1}(A^k x) = V^{-1}(V\Lambda V^{-1})^k x = \Lambda^k (V^{-1}x).$$

## 3.3. Χαρακτηριστικό Πολυώνυμο Πίνακα

Η εξίσωση ιδιοτιμής για έναν πίνακα *A* είναι

$$Av = \lambda v$$

που είναι ισοδύναμο με

$$(A - \lambda I) v = 0,$$

όπου *I* είναι ο $n \times n$ ταυτοτικός πίνακας. Είναι ένα θεμελιώδες αποτέλεσμα της γραμμικής άλγεβρας ότι μια εξίσωση $Mv = 0$ έχει μια μη μηδενική λύση *v* αν και μόνο αν η ορίζουσα *det(M)* του πίνακα *M* είναι μηδέν. Άμεσο επακόλουθο είναι ότι οι ιδιοτιμές του πίνακα *A* είναι ακριβώς οι πραγματικοί αριθμοί $\lambda$ που ικανοποιούν την εξίσωση

$$det(A - \lambda I)=0.$$

Το αριστερό μέλος αυτής της εξίσωσης μπορούμε να το δούμε (χρησιμοποιώντας τον κανόνα Leibniz για την ορίζουσα) ως μία πολυωνυμική συνάρτηση της μεταβλητής $\lambda$. Ο βαθμός του πολυωνύμου αυτού, είναι n, όσο και η τάξη του πίνακα. Οι συντελεστές του εξαρτώνται απο τις εκχωρήσεις στον *A*, με μόνη διαφορά ότι ο όρος βαθμού *n* είναι πάντα $(-1)^n \lambda^n$. Το πολυώνυμο αυτό ονομάζεται **χαρακτηριστικό πολυώνυμο** του *A* και η παραπάνω εξίσωση, χαρακτηριστική εξίσωση του *A*.

Για παράδειγμα, $A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 4 \\ 0 & 4 & 9 \end{bmatrix}$

Το χαρακτηριστικό πολυώνυμο του ***A*** είναι

$$\det(A - \lambda I) = \det\left(\begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 4 \\ 0 & 4 & 9 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) = \det\begin{bmatrix} 2-\lambda & 0 & 0 \\ 0 & 3-\lambda & 4 \\ 0 & 4 & 9-\lambda \end{bmatrix}$$

που είναι $(2 - \lambda)\left[(3 - \lambda)(9 - \lambda) - 16\right] = \lambda^3 - 14\lambda^2 + 35\lambda - 22$

Οι ρίζες του πολυωνύμου αυτού είναι 2, 1, και 11. Πράγματι, αυτές οι τρεις είναι οι μόνες ιδιοτιμές του ***A***, που αντιστοιχούν στα ιδιοδιανύσματα [1,0,0]', [0,2,-1]'και [0,1,2]' (ή κάθε μη μηδενικό πολλαπλάσιο τους).

## 3.4. Υπολογισμός των Ιδιοτιμών Πίνακα

Οι ιδιοτιμές ενός πίνακα *A* μπορούν να προσδιοριστούν βρίσκοντας τις ρίζες του χαρακτηριστικού πολυωνύμου. Σαφείς αλγεβρικοί τύποι για τις ρίζες ενός πολυωνύμου υπάρχουν μόνο αν ο βαθμός του *n* είναι 4 ή μικρότερος. Σύμφωνα με το θεώρημα Abel-Ruffini δεν υπάρχει γενικός, σαφής και ακριβής αλγεβρικός τύπος που να υπολογίζει τις ρίζες ενός πολυωνύμου βαθμού 5 ή μεγαλύτερου.

Υπογραμμίζουμε **τα βήματα για την επίλυση του προβλήματος ιδιοτιμών**:
1. Υπολόγισε την ορίζουσα $A-\lambda I$. Όταν αφαιρεθεί το $\lambda$ από τα διαγώνια στοιχεία, η ορίζουσα αυτή είναι ένα πολυώνυμο βαθμού *n*.
2. Βρες τις ρίζες αυτού τον πολυωνύμου. Οι *n* ρίζες είναι οι ιδιοτιμές.
3. Για κάθε μία ιδιοτιμή, λύσε το σύστημα $(A-\lambda I)x = 0$. Επειδή η ορίζουσα είναι μηδέν, υπάρχουν λύσεις διαφορετικές της $x = 0$. Αυτές είναι τα ιδιοδιανύσματα.

Για τους περισσότερους πίνακες, το πρόβλημα ιδιοτιμών είναι, χωρίς αμφιβολία, υπολογιστικά δυσκολότερο από το $Ax = b$. Η ακριβής λύση γραμμικών συστημάτων προέκυψε με ένα πεπερασμένο πλήθος βημάτων και σε πεπερασμένο χρόνο. (Ή, ισοδύναμα διατυπωμένο, ο κανόνας του Cramer έδωσε έναν ακριβή τύπο για την εν λόγω λύση.) Στην περίπτωση των ιδιοτιμών δεν μπορεί να υπάρχουν ούτε τέτοια βήματα ούτε τέτοιος τύπος,

Προκύπτει ότι κάθε πολυώνυμο βαθμού n είναι το χαρακτηριστικό πολυώνυμο κάποιου συνοδευτικού πίνακα τάξης *n*. Επομένως, **για πίνακες τάξης 5 ή παραπάνω, οι ιδιοτιμές και τα ιδιοδιανύσματα δεν μπορούν να προκύψουν από έναν ειδικό αλγεβρικό τύπο, και πρέπει συνεπώς να υπολογιστούν με προσεγγιστικές μεθόδους**.

Θεωρητικά, οι συντελεστές του χαρακτηριστικού πολυωνύμου μπορούν να υπολογιστούν ακριβώς, αφού είναι αθροίσματα γινομένων από στοιχεία πινάκων, και υπάρχουν αλγόριθμοι που μπορούν να βρουν όλες τις ρίζες ενός πολυωνύμου αυθαίρετου βαθμού με όση ακρίβεια ζητηθεί. Παρόλαυτα, η προσέγγιση αυτή δεν είναι εφικτή στην πράξη, επειδή οι συντελεστές θα επηρεάζονταν από αναπόφευκτα λάθη στρογγυλοποίησης, και οι ρίζες ενός πολυωνύμου μπορει να είναι μια εξαιρετικά ευαίσθητη συνάρτηση των συντελεστών. (Οπως για παράδειγμα το πολυώνυμο του Wilkinson).

Αποτελεσματικές, ακριβείς μέθοδοι υπολογισμού ιδιοτιμών και ιδιοδιανυσμάτων τυχαίων πινάκων δεν ήταν γνωστές, μέχρι την επινόηση του αλγορίθμου QR το 1961. Συνδυάζοντας τον μετασχηματισμό Χαουζχόλντερ με την LU αποσύνθεση, παίρνουμε αποτελέσματα σε έναν αλγόριθμο με καλύτερη σύγκλιση από τον αλγόριθμο QR. Για μεγάλους ερμιτιανούς αραιούς πίνακες, ο αλγόριθμος Lanczos είναι ένα παράδειγμα μιας αποτελεσματικής επαναληπτικής μεθόδου για να υπολογίσουμε ιδιοτιμές και ιδιοδιανύσματα, ανάμεσα σε αρκετές άλλες πιθανότητες.

## 3.5. Η Χρησιμότητα των Ιδιοτιμών

Οι ιδιοτιμές μπορούν να φανούν ιδιαίτερα χρήσιμες στις ακόλουθες περιπτώσεις [TREFETHEN2005] :

**Διαγωνοποίηση και διαχωρισμός μεταβλητών: χρήση των ιδιοσυναρτήσεων ως βάσης**. Ένα πράγμα που μπορούν να επιτύχουν οι ιδιοτιμές είναι η αποσύνδεση (decoupling) ενός προβλήματος που περιλαμβάνει διανύσματα ή συναρτήσεις σε μια συλλογή προβλημάτων που περιλαμβάνουν βαθμωτές τιμές, κάτι που μπορεί να κάνει ευκολότερους τους επόμενους υπολογισμούς.

**Συντονισμός: αυξημένη απόκριση σε επιλεγμένες εισόδους.**Η ανάλυση του φαινομένου του συντονισμού, ίσως το πιο οικείο στο πλαίσιο των δονούμενων χορδών, των τυμπάνων και των μηχανικών δομών. Κάθε επισκέπτης σε μουσεία επιστήμης έχει δει επιδείξεις που δείχνουν ότι ορισμένα συστήματα ανταποκρίνονται κατά προτίμηση σε δονήσεις σε ειδικές συχνότητες. Αυτές οι συχνότητες είναι οι ιδιοτιμές του γραμμικού ή γραμμικοποιημένου τελεστή που διέπει το εν λόγω σύστημα και η μορφή της απόκρισης σχετίζεται με τις αντίστοιχες ιδιοσυναρτήσεις. Παραδείγματα συντονισμού είναι γνωστά: στρατιώτες που αποσυντονίζουν το βήμα τους καθώς περνούν γέφυρες, σοπράνο της οποίας το ψηλό Ε σπάει τα παράθυρα, το ραδιόφωνο ΑΜ, όπου το σήμα από έναν μακρινό σταθμό επιλέγεται από μια θάλασσα θορύβου από ένα καλά συντονισμένο κύκλωμα συντονισμού, και τον κοχλία του ανθρώπινου αυτιού, του οποίου η βασική μεμβράνη αντηχεί κατά προτίμηση σε διαφορετικές τοποθεσίες ανάλογα με τη συχνότητα της εισόδου του ήχου.

**Ασυμπτωτική ανάλυση και σταθερότητα: η κυρίαρχη απόκριση σε γενικές εισόδους**. Τι θα συμβεί όσο περνάει ο χρόνος (ή στο ακραίο, $t \to \infty$) σε ένα σύστημα που έχει βιώσει κάποια περισσότερο ή λιγότερο τυχαία διαταραχή; Το πρόβλημα θερμότητας του Fourier δίνει και πάλι ένα παράδειγμα: Όποιο κι αν είναι το σχήμα της αρχικής κατανομής θερμοκρασίας, τα υψηλότερα ημιτονοειδή κύματα φθίνουν γρηγορότερα από τα χαμηλότερα, και επομένως σχεδόν οποιαδήποτε αρχική κατανομή θα μοιάζει τελικά με το ημίτονο του μισού μήκους κύματος με μηδενικά ακριβώς στα δύο άκρα του διαστήματος. Παρόμοια ερωτήματα προκύπτουν στη θεωρία ελέγχου και στην αριθμητική ανάλυση, όπου ο χρόνος είναι διακριτός και η σταθερότητα εξαρτάται τις τιμές των ιδιοτιμών. Τα προβλήματα σύγκλισης των επαναλήψεων πινάκων στην αριθμητική ανάλυση σχετίζονται επίσης, με το ποσοστό σύγκλισης να καθορίζεται από το πόσο κοντά είναι ορισμένες ιδιοτιμές στο μηδέν.

## 3.6. Διαταραχές Ιδιοτιμών

Το πρόβλημα των διαταραχών των ιδιοτιμών ενός πίνακα: Δεδομένου ενός $\nu \times \nu$ πίνακα $A$ και ενός $\nu \times \nu$ πίνακα διαταραχής $E$, ορίζουμε τον $\bar{A} = A + E$ και μελετάμε τις σχέσεις που συνδέουν τις ιδιοτιμές των πινάκων $\bar{A}$ και $A$.

Οι διαταραχές $\bar{A} = A+E$ όπου $||\varepsilon|| < \varepsilon$ για κάποιο $\varepsilon > 0$ ενός τετραγωνικού πίνακα $A \in \mathbb{C}^{\nu x \nu}$ καθώς και οι ιδιοτιμές τους αποτελούν τη βάση για τον ορισμό και τη μελέτη του ψευδοφάσματος ενός τετραγωνικού πίνακα και γι' αυτό στο παρόν κεφάλαιο θα μελετήσουμε το πρόβλημα διαταραχών ιδιοτιμών, τη σχέση δηλαδή των ιδιοτιμών των διαταραχών με αυτές του αρχικού πίνακα. Οι ιδιοτιμές των διαταραχών τείνουν να συγκεντρώνονται γύρω από τις ιδιοτιμές του αρχικού πίνακα. Η παρατήρηση αυτή είναι η βασική ιδέα για τον ορισμό που ψευδοφάσματος που θα δούμε παρακάτω.

<u>Θεώρημα.</u> Έστω $A \in \mathbb{C}$ μία ιδιοτιμή ενός τυχαίου πίνακα $A$, αλγεβρικής πολλαπλότητας $m$. Τότε, για κάθε νόρμα πινάκων $||\circ||$ και για κάθε $\delta > 0$, υπάρχει $\varepsilon > 0$ τέτοιο ώστε για κάθε πίνακα $E$ με $||\varepsilon|| < \varepsilon$, ο δίσκος $\Delta(\lambda, \delta)$ να περιέχει ακριβώς $m$ ιδιοτιμές του πίνακα $A$ (λαμβάνοντας υπόψη και τις πολλαπλότητες).

Έτσι, επιβεβαιώνεται η συνέχεια των διαταραγμένων ιδιοτιμών ως προς τα στοιχεία του $E$. Επομένως, οι ιδιοτιμές των διαταραχών τείνουν να συγκεντρώνονται γύρω από τις ιδιοτιμές του αρχικού πίνακα.

## 3.7. Ψευδοφάσμα πίνακα

Οι διαταραχές ενός πίνακα και κατ' επέκταση των ιδιοτιμών του αποτελούν τη βάση για τον ορισμό του ψευδοφάσματος ενός πίνακα $A \in \mathbb{C}^{vxv}$ , $\sigma_\varepsilon(A)$, για δεδομένο $\varepsilon > 0$. Θα δούμε πως το σύνολο αυτό "διαστέλεται" καθώς η παράμετρος $\varepsilon$ αυξάνει. Θα μελετήσουμε τις ιδιότητές του σε σχέση με τις ιδιότητες του $A$. Όπου απαιτείται, το ψευδοφάσμα ορίζεται για νόρμες πινάκων που επάγονται από νόρμες διανυσμάτων.

**Για ένα $\varepsilon > 0$, το ψευδοφάσμα ενός $v$ x $v$ μιγαδικού πίνακα $A$ είναι το σύνολο όλων των ιδιοτιμών όλων των πινάκων που απέχουν από τον $A$ απόσταση (κατά νόρμα) μικρότερη ή ίση του $\varepsilon$.**

Ως **φάσμα** $\sigma(A)$ ενός τετραγωνικού πίνακα $A$, ορίζεται το σύνολο των ιδιοτιμών του. Είναι φανερό ότι το φάσμα αποτελεί ειδική περίπτωση ψευδοφάσματος, για $\varepsilon = 0$.

Ο πίνακας $(\lambda - A)^{-1}$ είναι γνωστός ως ο επιλύων πίνακας του $A$ στο $\lambda$.

Θα δώσουμε τώρα 4 ισοδύναμους τυπικούς ορισμούς του ψευδοφάσματος:

1. $\sigma_\varepsilon(A) = \{\lambda \in \mathbb{C} : \ ||(\lambda I_v - A)^{-1}|| \ \geq \varepsilon^{-1}\}$. Το $\varepsilon$-ψευδοφάσμα είναι το ανοιχτό υποσύνολο του μιγαδικού επιπέδου που οριοθετείται από την $\varepsilon^{-1}$καμπύλη στάθμης της νόρμας του επιλύοντα πίνακα.
2. $\sigma_\varepsilon(A) = \{\lambda \in \mathbb{C} : \lambda \in \sigma(A + E)$ για κάποιον $E \in \mathbb{C}^{vxv}$ $με$ $||E|| \leq \varepsilon\}$. Το ε-ψευδοφάσμα είναι το σύνολο των αριθμών που είναι ιδιοτιμές κάποιου διαταραγμένου πίνακα $A + E$ με $||E|| < \varepsilon$.
3. $\sigma_\varepsilon(A) = \{\lambda \in \mathbb{C} : || \ (\lambda \ I_v - A) \ v|| < \varepsilon$ για κάποιο $v \in \mathbb{C}^v$ $με$ $||v|| = 1\}$.Ο αριθμός $\lambda$ είναι μια $\varepsilon$-ψευδοιδιοτιμή του $A$, και το $v$ είναι ένα αντίστοιχο $\varepsilon$-ψευδοιδιοδιάνυσμα. Το $\varepsilon$-ψευδοφάσμα είναι το σύνολο των $\varepsilon$-ψευδοιδιοτιμών.
4. $\sigma_\varepsilon(A) = \{\lambda \in \mathbb{C} : s_{min}(\lambda \ I_v - A) < \varepsilon\}$, όταν αναφερόμαστε στη νόρμα $|| \ o \ ||_2$, συμβολίζοντας με $s_{min}(\cdot)$ την ελάχιστη ιδιάζουσα τιμή ενός πίνακα.

Στην περίπτωση που $\lambda \in \sigma(A)$, θεωρούμε ότι $||(\lambda I_v — A)^{-1}|| = \infty$.

Το φάσμα περιέχεται στο $\varepsilon$-ψευδοφάσμα για κάθε $\varepsilon > 0$. Έτσι, σε αντίθεση με το φάσμα, τα ψευδοφάσματα εξαρτώνται από τη νόρμα. Εκ πρώτης όψεως αυτή η έλλειψη αμετάβλητης νόρμας μπορεί να φαίνεται ως ελάττωμα στην ιδέα των ψευδοφασμάτων, και σίγουρα συνέβαλε στο γεγονός ότι η ανάπτυξη μιας θεωρίας μη κανονικότητας έχει μείνει πολύ πίσω από την ανάπτυξη της τυπικής φασματικής θεωρίας. (Τα ψευδοφάσματα είναι μια ιδέα της ανάλυσης, ενώ οι ιδιοτιμές ανήκουν στην άλγεβρα.) Ωστόσο, αυτό που πραγματικά χρειάζεται να γνωρίζει κανείς για ένα εφαρμοσμένο πρόβλημα εξαρτάται συνήθως από τη νόρμα. [TREFETHEN2005]

## 3.8. Σημασία του Ψευδοφάσματος Πίνακα

Η πλειονότητα των γνωστών εφαρμογών της ανάλυσης ιδιοτιμών περιλαμβάνει πίνακες ή τελεστές που είναι κανονικοί ή κοντά στο κανονικό, με ιδιοσυναρτήσεις ορθογώνιες ή σχεδόν. Οι γνωστές μηχανικές ταλαντώσεις διέπονται, για παράδειγμα, από κανονικούς τελεστές, όπως και οι ταλαντώσεις της κβαντικής μηχανικής, τουλάχιστον στην τυπική τους διατύπωση. Κατά συνέπεια, η διαίσθησή μας για τις ιδιοτιμές έχει διαμορφωθεί από την κανονική περίπτωση. Δύο αιώνες επιτυχιών έχουν δημιουργήσει σιγουριά ότι η ιδέα της ιδιοτιμής είναι τόσο ισχυρή στην πράξη όσο και θεμελιώδης ως προς την ιδέα. Οι περισσότερες από αυτές τις επιτυχίες περιλαμβάνουν προβλήματα που διέπονται από κανονικούς ή σχεδόν κανονικούς τελεστές.

Υπάρχει όμως μια κατηγορία προβλημάτων για τα οποία οι μέθοδοι ιδιοτιμών ενδέχεται να αποτύχουν: προβλήματα που αφορούν πίνακες ή τελεστές για τους οποίους ο πίνακας $V^{-1}$, στην αποσύνθεση ιδιοτιμών

$$A = V \Lambda V^{-1}$$

εάν υπάρχει, περιέχει πολύ μεγάλες τιμές:

$$|| V^{-1}|| \gg 1.$$

Αυτό συχνά σημαίνει ότι είναι εκθετικά μεγάλο σε σχέση με μια παράμετρο.

Δεν είναι προφανές στην αρχή εάν η ιδέα των ψευδοφασμάτων εξυπηρετεί κάποιο σκοπό. Δεν είναι το $||(z - A)^{-1}||$ μεγάλο, ακριβώς όταν το $z$ είναι κοντά σε μια ιδιοτιμή του $A$; Για έναν κανονικό πίνακα, όταν $|| \cdot || = || \cdot ||_2$, αυτή η διαίσθηση είναι σωστή. Ας το δούμε σχηματικά.

(α) κανονικός πίνακας (β) μη κανονικός πίνακας

Η γεωμετρία των ψευδοφασμάτων: σχηματική όψη.
Σε κάθε διάγραμμα, τα περιγράμματα αντιπροσωπεύουν το σύνορο του $\sigma_\varepsilon(A)$ για δύο τιμές του $\varepsilon$.

Η σημασία των ψευδοφασμάτων προκύπτει για πίνακες που απέχουν πολύ από το κανονικό, για τους οποίους το $||(z - A)^{-1}||$ μπορεί να είναι μεγάλο ακόμη και όταν το $z$ είναι μακριά από το φάσμα.

## 3.9. Παραδείγματα για το Ψευδοφάσμα Πίνακα

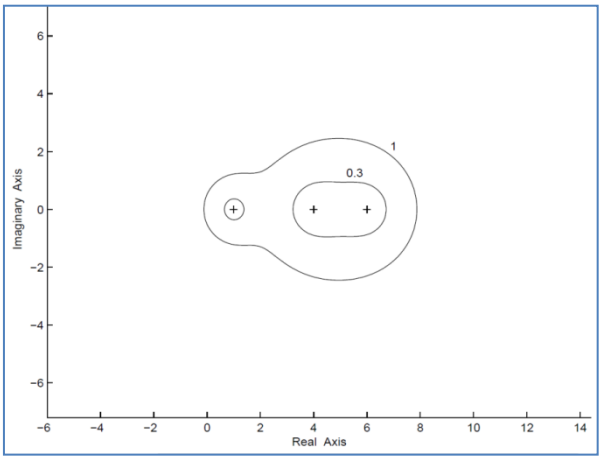
Έστω ο διαγωνοποιήσιμος πίνακας

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix} = S\Lambda S^{-1} = S \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{bmatrix} S^{-1},$$

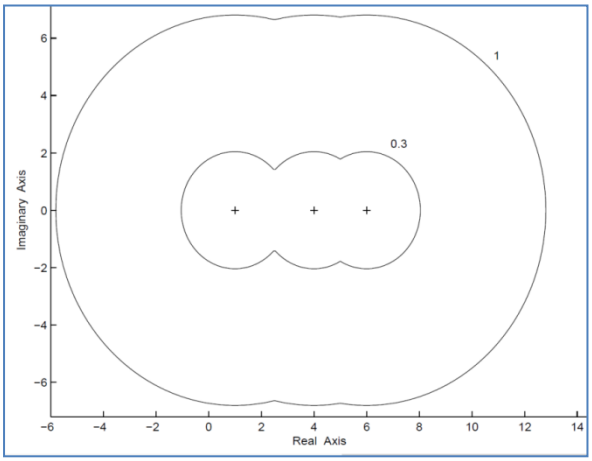
όπου ο πίνακας μετασχηματισμού

$$S = \begin{bmatrix} 1 & 0.5547 & 0.5108 \\ 0 & 0.8321 & 0.7982 \\ 0 & 0 & 0.3193 \end{bmatrix}$$

έχει βαθμό κατάστασης $k_2(S) = ||S||_2 \, ||S^{-1}||_2 = 6.8066$.




Τα ψευδοφάσματα του $A$ για $\varepsilon = 0.3$ και 1.

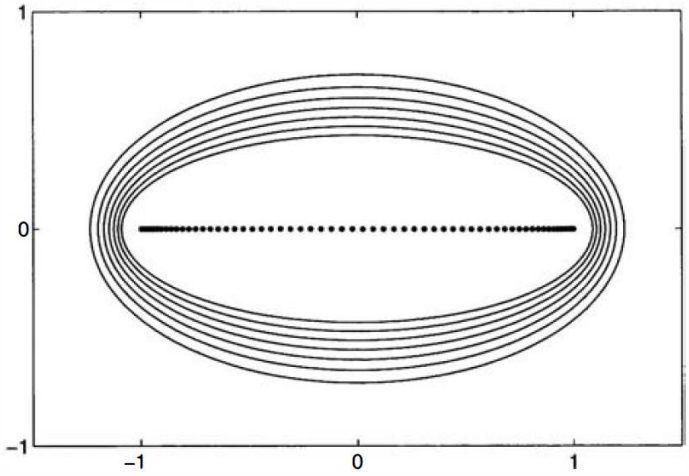



Τα ψευδοφάσματα του $A$ για $\varepsilon = 0.3$ $k_2(S) = 2.0420$ και $\varepsilon = k_2(S) = 6.8066$.

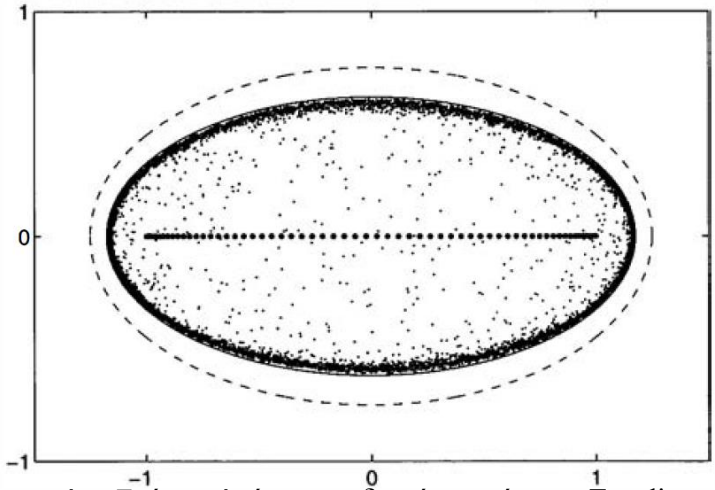
Συγκρίνοντας τα ψευδοφάσματα, βλέπουμε ότι $S_{0.3}(A) \subset S_{2.0420}(\Lambda)$ και $S_1(A) \subset S_{6.8066}(\Lambda)$. Αν και δεν φαίνεται με την πρώτη ματιά, τα ψευδοφάσματα του διαγώνιου πίνακα $\Lambda$ είναι η ένωση τριών κυκλικών δίσκων.

Οι τριδιαγώνιοι πίνακες Toeplitz και οι διαταραχές τέτοιων πινάκων προκύπτουν σε πολλές εφαρμογές, συμπεριλαμβανομένης της επίλυσης συνήθων και μερικών διαφορικών εξισώσεων,ανάλυση χρονοσειρών κλπ. Είναι επομένως σημαντικό να κατανοήσουμε τις ιδιότητές τους.Οι ιδιοτιμές πραγματικών και μιγαδικών τριδιαγώνιων πινάκων Toeplitz μπορεί να είναι πολύ ευαίσθητες στις διαταραχές του πίνακα.

Ας εξετάσουμε τον τριδιαγώνιο πίνακα Toeplitz $\mathbf{A} = \begin{pmatrix} 0 & 1 & & & \\ \frac{1}{4} & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{4} & 0 & 1 \\ & & & \frac{1}{4} & 0 \end{pmatrix} \in \mathbb{C}^{N\times N}$




Όρια ψευδοφασμάτων $\sigma_\varepsilon(A)$, $\varepsilon = 10\text{-}2, 10\text{-}3, ..., 10\text{-}8$,για τον παραπάνω τριδιαγώνιο πίνακα Toeplitz με διάσταση $N = 64$.Οι ιδιοτιμές σημειώνονται με συμπαγείς τελείες.




Υπέρθεση ιδιοτιμών 100 πινάκων $A + E$, όπου $A$ είναι ο τριδιαγώνιος πίνακας Toeplitz με διάσταση $N = 64$ και κάθε $E$ είναι ένας τυχαίος πίνακας με $||E|| = 10^{-3}$. Οι ιδιοτιμές του $A$ είναι πραγματικές (μεγαλύτερες τελείες), αλλά η διαταραχή που εισάγεται από το $E$ τις μετακινεί πολύ στο μιγαδικό επίπεδο, κοντά στην έλλειψη με $|z| = (0.001)^{1/64}$ (συμπαγής καμπύλη). Η διακεκομμένη έλλειψη αντιστοιχεί σε $N \to \infty$ και $|z| = 1$.

## 3.10. Ψευδοφάσμα: Ιστορική Αναδρομή

Τα ψευδοφάσματα είναι ένα ισχυρό εργαλείο για την κατανόηση της συμπεριφοράς πολύπλοκων συστημάτων. Μπορούν να χρησιμοποιηθούν για την οπτικοποίηση της ευαισθησίας ενός συστήματος σε διαταραχές, για τη λήψη ποσοτικών ορίων στη συμπεριφορά του συστήματος και για τον εντοπισμό ασταθών τρόπων λειτουργίας. Τα ψευδοφάσματα έχουν χρησιμοποιηθεί σε μια ευρεία ποικιλία εφαρμογών, συμπεριλαμβανομένης της σύγκλισης επαναλήψεων μη συμμετρικών πινάκων, της ανάλυσης σφαλμάτων προς τα πίσω των αλγορίθμων ιδιοτιμών, της σταθερότητας των φασματικών μεθόδων και της αναγνώρισης ασταθών τρόπων λειτουργίας. Έχουν επίσης χρησιμοποιηθεί για τη λήψη σημαντικών αποτελεσμάτων στη φασματική θεωρία και τις φασματικές ιδιότητες των ζωνοποιημένων πινάκων Toeplitz.

Τα ψευδοφάσματα έχουν τις ρίζες τους στη θεωρία των διαταραχών των γραμμικών τελεστών και παρέχουν πληροφορίες για το πώς οι ιδιοτιμές ενός πίνακα μπορούν να αλλάξουν κάτω από μικρές διαταραχές.

Ο J. M. Varah, στη διατριβή του το 1967 στο Πανεπιστήμιο του Στάνφορντ, εισήγαγε την έννοια της s-pseudoeigenvalue, γνωστής και ως r-προσεγγιστική ιδιοτιμή. Όρισε το 2-norm E-pseudospectrum ως προς την ελάχιστη μοναδική τιμή smin (A - )..), δίνοντάς του το όνομα E-spectrum και τον συμβολισμό S.:(A). Ο Varah τόνισε ότι για μια μη φυσιολογική μήτρα, τα ψευδοφάσματα μπορεί να είναι πολύ διαφορετικά από το φάσμα.

Ο H. J. Landau εισήγαγε ανεξάρτητα E-pseudoeigenvalues με το όνομα E-proximate eigenvalues το 1975. Η εργασία του Landau εφάρμοσε αυτή την έννοια στη θεωρία των πινάκων Toeplitz και των σχετικών ολοκληρωτικών τελεστών.

Στη δεκαετία του 1980, σημαντικές συνεισφορές έγιναν από τον S. K. Godunov και τους συναδέλφους του στο Novosibirsk. Μελέτησαν πώς η μη κανονικότητα επηρεάζει την αριθμητική σταθερότητα των διακριτικών διαφορικών εξισώσεων.

Η επιδραστική εργασία του L. N. Trefethen «Pseudospectra of Matrices», που δημοσιεύτηκε το 1992, παρουσίασε την ιδέα των ψευδοφασμάτων και παρουσίασε δεκατρία παραδείγματα. Αυτό το έργο σηματοδότησε ένα σημείο καμπής, μετά το οποίο η έννοια των ψευδοφασμάτων άρχισε να κερδίζει ευρεία αναγνώριση.

Οι D. Hinrichsen και A. J. Pritchard, στα μέσα της δεκαετίας του 1980, έγραψαν αρκετές εργασίες σχετικά με την ακτίνα σταθερότητας μιας μήτρας. Σε μια εργασία του 1992, εισήγαγαν τον όρο σύνολο φασματικών τιμών και τον συμβολισμό dA, p) για να δηλώσουν το πραγματικό δομημένο c-ψευδοφάσμα ενός μη κανονικού πίνακα.

Ο L. N. Trefethen κατέληξε στο συμπέρασμα ότι τα ψευδοφάσματα έχουν εφευρεθεί ανεξάρτητα τουλάχιστον πέντε φορές από διαφορετικούς ερευνητές: J. M. Varah, H. J. Landau, S. K. Godunov, L. N. Trefethen και (D. Hinrichsen και A. J. Pritchard)!

## 3.11. Πρακτικοί Αλγόριθμοι Σχεδίασης του Ψευδοφάσματος

Συχνά, στις πρακτικές εφαρμογές η έννοια του ψευδοφάσματος είναι πολύ πιο χρήσιμη από αυτή του φάσματος, διότι εμπεριέχει τις ιδιοτιμές αλλά και τη συμπεριφορά τους για μικρές διαταραχές του πίνακα. Εκτός αυτού, η συμπεριφορά του πίνακα κοντά στις ιδιοτιμές είναι πολλές φορές απρόβλεπτη ακόμα και για σχετικά απλούς πίνακες.

Το ε-ψευδοφάσμα ενός τετραγωνικού πίνακα $A$ είναι ένα σύνολο του μιγαδικού επιπέδου το οποίο περιλαμβάνει τις ιδιοτιμές του πίνακα καθώς και τιμές οι οποίες είναι «σχεδόν» ιδιοτιμές. Πρόκειται για μια επέκταση του φάσματος (του συνόλου δηλαδή των ιδιοτιμών) για να καλύψει σταδιακά ολόκληρο το μιγαδικό επίπεδο και αποτελείται από τις ιδιοτιμές των διαταραχών του $A$, δηλαδή των πινάκων $A + E$, όπου $E$ ένας πίνακας με $||E|| < \varepsilon$ , για κάποιον μη αρνητικό αριθμό $\varepsilon$.Το φάσμα αποτελεί ειδική περίπτωση του ψευδοφάσματος για $\varepsilon = 0$.

Πού «ζει» ένας πίνακας $A$ στο μιγαδικό επίπεδο; Εάν ο $A$ είναι κανονικός, τότε το φάσμα $\sigma(A)$ είναι μια ικανοποιητική απάντηση σε αυτήν την ερώτηση για σχεδόν κάθε σκοπό. Η οικογένεια των ψευδοφασμάτων $\sigma_\varepsilon(A)$ είναι μια προσπάθεια να δοθεί μια ικανοποιητική απάντηση στην περίπτωση που ο $A$ δεν είναι κανονικός, ίσως πολύ μακριά από τον κανονικό. Δεν είναι τέλεια απάντηση. Τα ψευδοφάσματα στερούνται την απλότητα των φασμάτων, ωστόσο, παρά την πολυπλοκότητά τους, δεν παρέχουν ακριβείς απαντήσεις στα ερωτήματα που θα ήθελε κανείς να κάνει σχετικά με τη συμπεριφορά του $A$. Παρέχουν κατά προσέγγιση απαντήσεις, ωστόσο, με τη μορφή ορίων που συχνά είναι αρκετά στενά. Παρέχουν μια υπενθύμιση ότι οι ιδιοτιμές που είναι ευαίσθητες σε διαταραχές μπορεί να έχουν περιορισμένη σημασία για τον προσδιορισμό της συμπεριφοράς του $A$. Τέλος, παρέχουν μια ελκυστική γεωμετρική ερμηνεία της μη κανονικότητας. Το γεγονός είναι ότι οι μη κανονικοί πίνακες και οι τελεστές δεν ζουν στο μιγαδικό επίπεδο, αλλά μπορεί κανείς να κάνει μια καλή αρχή στην πρόβλεψη της συμπεριφοράς τους, εάν, εκτός από τον συνήθη υπολογισμό των ιδιοτιμών, σχεδιάσει μερικές γραμμές περιγράμματος της επιλύουσας συνάρτηση ή των ιδιοτιμών μερικών τυχαία διαταραγμένων πινάκων.[TREFETHEN2005]

Υπάρχουν άπειροι τέτοιοι πίνακες $E$ για κάθε τιμή του $\varepsilon$, γεγονός που δηλώνει ότι το ψευδοφάσμα δεν μπορεί να προσδιοριστεί επαρκώς χωρίς τη βοήθεια υπολογιστή.Έχουν αναπτυχθεί αρκετοί αλγόριθμοι για τον υπολογισμό κυρίως του συνόρου του ψευδοφάσματος, το οποίο είναι το σύνολο

$$\partial\Lambda_\varepsilon \subseteq \left\{ z \in \mathbb{C} : \left\| (z\mathbf{I} - \mathbf{A})^{-1} \right\| = \varepsilon^{-1} \right\} \equiv \left\{ z \in \mathbb{C} : s_{\min}(z\mathbf{I} - \mathbf{A}) = \varepsilon \right\} \qquad (*)$$

όπου το σύνολο $\left\{ z \in \mathbb{C} : \left\| (z\mathbf{I} - \mathbf{A})^{-1} \right\| = \varepsilon^{-1} \right\} \setminus \partial\Lambda_\varepsilon$

αποτελείται από το πολύ $n$ σημεία και άρα θεωρούμε, χωρίς βλάβη της γενικότητας ότι τα σύνολα της σχέσης (*) ταυτίζονται μεταξύ τους.

Οι αλγόριθμοι που θα παρουσιάσουμε έχουν στόχο την γραφική σχεδίαση του ψευδοφάσματος, την εύρεση δηλαδή του συνόρου του, $\partial\Lambda_\varepsilon$, για κάθε τιμή του ε και κατατάσσονται σε δύο βασικές κατηγορίες, τους αλγόριθμους Grid και τους αλγόριθμους Path-Following.

# 4. Ο Αλγόριθμος GRID

## 4.1. Περιγραφή του Αλγορίθμου GRID

Ο απλούστερος της κατηγορίας είναι ο αρχικός αλγόριθμος GRID. Για την εύρεση του ψευδοφάσματος ενός πίνακα $A \in \mathbb{C}^{v \times v}$, ο αλγόριθμος GRID επιλέγει ένα χωρίο του μιγαδικού επιπέδου, και κατασκευάζει ένα πλέγμα $\Omega$ (grid) μέσα στο χωρίο αυτό, που αποτελείται από σημεία z του μιγαδικού επιπέδου. Στη συνέχεια υπολογίζει την $(zI - A)$ για κάθε $z \in \Omega$ και αποφαίνεται εάν το συγκεκριμένο z ανήκει στο ψευδοφάσμα με βάση τον Ορισμό:

$$\sigma_\varepsilon(A) = \{\lambda \in \mathbb{C} : s_{min}(\lambda I_v - A) < \varepsilon\} ,$$

αναφερόμαστε στη νόρμα $\|$ ο $\|_2$, συμβολίζοντας με $s_{min}(\cdot)$ την ελάχιστη ιδιάζουσα τιμή ενός πίνακα.

Από τις μεθοδολογίες **επιλογής του χωρίου $\Omega$** επιλέγουμε τη χρήση του αριθμητικού πεδίου του πίνακα (Field of Values). Ως αριθμητικό πεδίο (Field of Values, FoV) του $A$ ορίζεται το σύνολο των πηλίκων Rayleigh:

$$Fov(A) = \left\{\frac{z^* Az}{z^* z}\right\}, \qquad z \in C^n, \; z \neq 0$$

Το παραπάνω σύνολο είναι κυρτό και αποτελεί το ελάχιστο κυρτό σύνολο που περιέχει το φάσμα του πίνακα. Η χρήση του αριθμητικού πεδίου είναι μια από τις πιο αποδοτικές μεθόδους για την επιλογή του χωρίου $\Omega$ καθώς προσεγγίζει κατά πολύ το ψευδοφάσμα. Ένα λάθος αρχικό χωρίο μπορεί να οδηγήσει το $\Omega$ να μην περιέχει ολόκληρο το ψευδοφάσμα. Δεν είναι, όμως, δυνατό να ξέρουμε εκ των προτέρων ποια είναι η καλύτερη επιλογή του χωρίου, διότι αυτό εξαρτάται από τη μορφή και τη γεωμετρία του ψευδοφάσματος πάνω στο μιγαδικό επίπεδο που δεν είναι γνωστή εξ'αρχής.

Ο αριθμός των SVD παραγοντοποιήσεων αποτελεί το κύριο χρονικό κόστος του. Δεν μπορούν όμως να μειωθούν στη συγκεκριμένη υλοποίηση καθώς ο αλγόριθμος πρέπει να υπολογίζει την $(zI - A)$ για κάθε σημείο του πλέγματος.

Ο αλγόριθμος GRID είναι μπορεί εύκολα να εκμεταλλευτεί πολλαπλούς επεξεργαστές. Οι υπολογισμοί σε κάθε σημείο του πλέγματος είναι ανεξάρτητοι, και έτσι η εργασία μπορεί να κατανεμηθεί σε όλους τους επεξεργαστές με ελάχιστη ανάγκη για επικοινωνία μεταξύ τους.Μπορούμε να υπολογίζουμε το ψευδοφάσμα του πίνακα για περισσότερες από μια τιμές του $\varepsilon$, δίνοντας την καθεμιά σε διαφορετικό επεξεργαστή. Βέβαια, η επιτυχής εκτέλεση προϋποθέτει το ψευδοφάσμα να ανήκει ολόκληρο στο αρχικό χωρίο που επιλέγει ο αλγόριθμος για κάθε τιμή του $\varepsilon$. Κάτι τέτοιο εξασφαλίζεται ξεκινώντας τον αλγόριθμο και υπολογίζοντας το χωρίο για τη μεγαλύτερη τιμή του $\varepsilon$, αφού το μέγεθος του ψευδοφάσματος μικραίνει όσο μικραίνει και το ε.

Ο αλγόριθμος GRID για την εύρεση του ψευδοφάσματος για μια τιμή του ε ξεκινά με την επιλογή του χωρίου $\Omega$ και στη συνέχεια, για κάθε σημείο του πλέγματος υπολογίζεται η $s_{min}$ $(zI - A)$,ώστε να προσδιοριστούν τα σημεία του $\Omega$ που ανήκουν στο $\partial\Lambda_\varepsilon$.

## 4.2. Ψευδοκώδικας

**ΕΙΣΟΔΟΣ:**

$A \in \mathbb{C}^{vxv}$

$\varepsilon > 0$

$N$, ο αριθμός των σημείων του πλέγματος σε κάθε άξονα

**ΒΗΜΑ 1:** Ορισμός των ορίων του πλέγματος και των βημάτων

$x_{min} = \lambda_{min}\ (\boldsymbol{A}_H) - \varepsilon\ ||\boldsymbol{A}||_2$

$x_{max} = \lambda_{max}\ (\boldsymbol{A}_H) - \varepsilon\ ||\boldsymbol{A}||_2$

$y_{min} = \lambda_{min}\ (\boldsymbol{A}_{SH}) - \varepsilon\ ||\boldsymbol{A}||_2$

$y_{max} = \lambda_{max}\ (\boldsymbol{A}_{SH}) - \varepsilon\ ||\boldsymbol{A}||_2$

$x_{step} = (x_{max} - x_{min}) / N$

$y_{step} = (y_{max} - y_{min}) / N$

**ΒΗΜΑ 2:**

Για κάθε σημείο $(x, y)$ του πλέγματος:

$z = x + iy$

Υπολογισμός της $s_{min}\ (zI - \boldsymbol{A})$

Τέλος Επανάληψης

Σχεδιάζονται τα σημεία εκείνα για τα οποία ισχύει $s_{min}\ (zI - \boldsymbol{A}) = \varepsilon$

**ΤΕΛΟΣ**

Για τον υπολογισμό του $\Lambda_\varepsilon$ για περισσότερα ε, ο αλγόριθμος εκτελεί την παραπάνω διαδικασία μόνο για το μεγαλύτερο εξ' αυτών και απλώς εκτελείται η τελευταία εντολή για κάθε τιμή[ΔΕΣΥΠΡΗ2018].

## 4.3. Κώδικας Matlab

Στηρίζεται σε κώδικα της διπλωματικής εργασίας [ΔΕΣΥΠΡΗ2018].

### file: grid 01 program.m

```
A = [   0.8147 0.0975 0.1576 0.1419 0.6557 ;
0.9058 0.2785 0.9706 0.4218 0.0357 ;
        0.1270 0.5469 0.9572 0.9157 0.8491 ;
        0.9134 0.9575 0.4854 0.7922 0.9340 ;
        0.6324 0.9649 0.8003 0.9595 0.6787
    ];

N = 10;

epsilons = [1  0.8  0.5  0.2  0.005];
[X, Y, Z, Vxmin, Vxmax, Vymin, Vymax, maxepsilon] = gridcalculate(A, epsilons, N);
gridplot(X, Y, Z, epsilons, Vxmin, Vxmax, Vymin, Vymax);
```

### file: gridplot.m

```
function gridplot(X, Y, Z, epsilons, Vxmin, Vxmax, Vymin, Vymax)
figure(1);   %% create a figure
    hold on;
    mycolour = "krgbm";

    [maxepsilon, ind] = max(epsilons);
    [axisxmin, axisxmax, axisymin, axisymax] =
grid_find_axis_limits(Vxmin(ind), Vxmax(ind), Vymin(ind), Vymax(ind));

axis([axisxmin axisxmax axisymin axisymax]);
          %% Change the axis limits so that the x-axis ranges from axisxmin to axisxmax and the y-axis ranges from axisymin to axisymax

    for i = 1:length(epsilons)
        %% Plotting the grid, black color
        tax = [Vxmin(i) Vxmax(i) Vxmax(i) Vxmin(i) Vxmin(i)];  %% x points for a rectangle
        tay = [Vymin(i) Vymin(i) Vymax(i) Vymax(i) Vymin(i)];  %% y points for a rectangle
        p = plot(tax, tay, mycolour(mod(i, length(mycolour)) + 1) );
%% plots Omega rectangle;    plot() plots a 2D line,
                                %% getting points from the corresponding values of the tax and tay vectors

        hold on;  %% hold on retains plots in the current axes so that new plots added do not delete existing plots

contour(X, Y, Z', [epsilons(i) epsilons(i)], mycolour(mod(i, length(mycolour)) + 1), "linewidth", 2) ;
             %% Display the contours of the Z', at Z' = epsilon
    end
    %%  pause;
    %%  print("grid_02_merged_matlab.jpg", "-djpg") ;   %% print the figure in the file
```

## file: gridcalculate.m

```
function [X, Y, Z, Vxmin, Vxmax, Vymin, Vymax, maxepsilon] = gridcalculate(A, epsilons, N)
%%  print_2d_array(A, "A");

n = length(A);
norma = norm(A,2);  %% norm(X, 2) returns the 2-norm or maximum singular value of matrix X,
 %% which is approximately max(svd(X)). This value is equivalent to norm(X) in Matlab, but not in Python numpy.
%%  fprintf("matrix A:   rows = %d,   2-norm = %f\n\n", n, norma);

Ah =  (A+A')/2;     %% A' is the transpose matrix
Ash = (A-A')/2;
%%  print_2d_array(Ah, "Ah");%%  print_2d_array(Ash, "Ash");

%% The eigenvalues of Ah and Ash in ascending order
EAh =  sort(real(eig(Ah)));     %% eig(A) returns a column vector containing
EAsh = sort(imag(eig(Ash)));    %% the eigenvalues of square matrix A.
%%  print_1d_array(EAh, "EAh");%%  print_1d_array(EAsh, "EAsh");

for metr = 1:length(epsilons)
    Vxmin(metr) = EAh(1) - epsilons(metr) * norma;    Vxmax(metr) = EAh(end) + epsilons(metr) * norma;
    Vymin(metr) = EAsh(1) - epsilons(metr) * norma;   Vymax(metr) = EAsh(end) + epsilons(metr) * norma;
    Vxstep(metr) = (Vxmax(metr) - Vxmin(metr)) / N;   Vystep(metr) = (Vymax(metr) - Vymin(metr)) / N;
    %% fprintf("epsilons(%d) = %f,   xmin = %f,   xmax = %f,   ymin= %f,   ymax = %f,    xstep = %f,     ystep = %f \n\n\n",
%%             metr, epsilons(metr), Vxmin(metr), Vxmax(metr), Vymin(metr), Vymax(metr), Vxstep(metr), Vystep(metr));
end

[maxepsilon, indexofmaxepsilon] = max(epsilons);
xmin = Vxmin(indexofmaxepsilon);xmax = Vxmax(indexofmaxepsilon);
ymin = Vymin(indexofmaxepsilon);ymax = Vymax(indexofmaxepsilon);
xstep = Vxstep(indexofmaxepsilon);ystep = Vystep(indexofmaxepsilon);
%%  fprintf("xmin = %f,    xmax = %f,    ymin= %f,    ymax = %f,    xstep = %f,     ystep = %f \n\n\n",
%%          xmin, xmax, ymin, ymax, xstep, ystep);

for I=1:N
    x = xmin + (I-1)*xstep;
    X(I)=x;
    for J=1:N
        y = ymin + (J-1)*ystep;
        Y(J)=y;
        z = x + i*y;
        S = svd(z*eye(n)-A);
        Z(I,J) = S(end);
    end
end

%%  fprintf("Results for max epsilon = %f \n\n", maxepsilon);
%%  print_1d_array(X, "X");    %%  print_1d_array(Y, "Y");
%%  print_2d_array(Z, "Z");%%  print_2d_array(Z', "Z'");

end
```

file:  grid find axis limits.m

```
function [axisxmin, axisxmax, axisymin, axisymax] = grid_find_axis_limits(xmin, xmax, ymin, ymax)


    if xmin < 0
        axisxmin = xmin * 1.2;
    elseif xmin == 0
        axisxmin = -1;
    else
        axisxmin = xmin / 1.2;
    end

    if ymin < 0
axisymin = ymin * 1.2;
elseif ymin == 0
axisymin = -1;
else
        axisymin = ymin / 1.2;
    end

    if xmax < 0
        axisxmax = xmax / 1.2;
    elseif xmax == 0
        axisxmax = 1;
    else
        axisxmax = xmax * 1.2;
    end

    if ymax < 0
        axisymax = ymax / 1.2;
    elseif ymax == 0
        axisymax = 1;
    else
        axisymax = ymax * 1.2;
    end

end
```

## file: print 1d array.m

```
function print_1d_array(A, name)
    fprintf("\n");
numberofelements = length(A);
printf("vector %s:    number of elements = %d \n", name, numberofelements);
    fprintf("\n");
    for i=1:numberofelements
fprintf("%12.4f", A(i));
        end
    fprintf("\n\n\n");
end
```

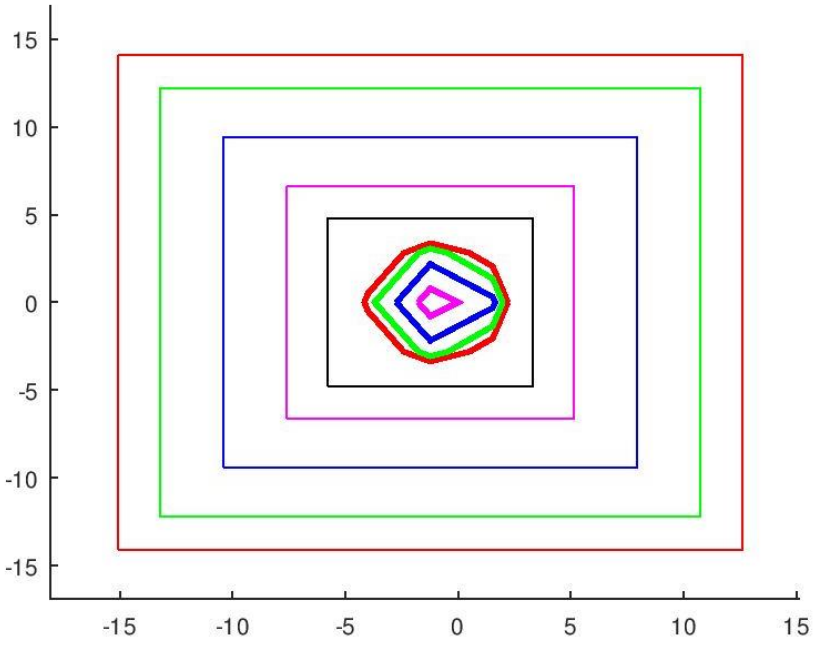
## file: print 2d array.m

```
function print_2d_array(A, name)
    fprintf("\n");
    [rows, cols] = size(A);
    totalelements = rows * cols;
printf("matrix %s:   rows = %d,   cols = %d,   total elements = %d \n", name, rows, cols, totalelements);
    fprintf("\n");
    for i=1:rows
        for j=1:cols
fprintf("%12.4f", A(i, j));
        end
        fprintf("\n");
    end
    fprintf("\n\n\n");
end
```

## 4.4. Αποτελέσματα του Κώδικα Matlab

```
matrix A:   rows = 5,   cols = 5,   total elements = 25

      0.8147      0.0975      0.1576      0.1419      0.6557
      0.9058      0.2785      0.9706      0.4218      0.0357
      0.1270      0.5469      0.9572      0.9157      0.8491
      0.9134      0.9575      0.4854      0.7922      0.9340
      0.6324      0.9649      0.8003      0.9595      0.6787

matrix A:   rows = 5,   2-norm = 3.312915

matrix Ah:   rows = 5,   cols = 5,   total elements = 25

      0.8147      0.5017      0.1423      0.5276      0.6441
      0.5017      0.2785      0.7588      0.6896      0.5003
      0.1423      0.7588      0.9572      0.7006      0.8247
      0.5276      0.6896      0.7006      0.7922      0.9467
      0.6441      0.5003      0.8247      0.9467      0.6787

matrix Ash:   rows = 5,   cols = 5,   total elements = 25

      0.0000     -0.4042      0.0153     -0.3857      0.0116
      0.4042      0.0000      0.2118     -0.2679     -0.4646
     -0.0153     -0.2118      0.0000      0.2151      0.0244
      0.3857      0.2679     -0.2151      0.0000     -0.0127
     -0.0116      0.4646     -0.0244      0.0127      0.0000

vector EAh:    number of elements = 5

     -0.4627     -0.0870      0.0740      0.7339      3.2631

vector EAsh:    number of elements = 5

     -0.7575     -0.3439      0.0000      0.3439      0.7575

epsilons(1) = 1.000000,    xmin = -3.775599,    xmax = 6.576030,    ymin= -4.070389,    ymax = 4.070389,    xstep = 1.035163,
ystep = 0.814078

epsilons(2) = 0.800000,    xmin = -3.113016,    xmax = 5.913447,    ymin= -3.407806,    ymax = 3.407806,    xstep = 0.902646,
ystep = 0.681561


epsilons(3) = 0.500000,    xmin = -2.119141,    xmax = 4.919572,    ymin= -2.413932,    ymax = 2.413932,    xstep = 0.703871,
ystep = 0.482786

epsilons(4) = 0.200000,    xmin = -1.125267,    xmax = 3.925698,    ymin= -1.420057,    ymax = 1.420057,    xstep = 0.505097,
ystep = 0.284011

epsilons(5) = 0.005000,    xmin = -0.479249,    xmax = 3.279680,    ymin= -0.774039,    ymax = 0.774039,    xstep = 0.375893,
ystep = 0.154808

xmin = -3.775599,    xmax = 6.576030,    ymin= -4.070389,    ymax = 4.070389,    xstep = 1.035163,    ystep = 0.814078
```

```
Results for max epsilon = 1.000000

vector X:    number of elements = 10

     -3.7756     -2.7404     -1.7053     -0.6701      0.3651      1.4002      2.4354      3.4705      4.5057      5.5409

vector Y:    number of elements = 10

     -4.0704     -3.2563     -2.4422     -1.6282     -0.8141      0.0000      0.8141      1.6282      2.4422      3.2563

matrix Z:   rows = 10,   cols = 10,   total elements = 100

      4.9637      4.4270      3.9755      3.6334      3.4210      3.3491      3.4210      3.6334      3.9755      4.4270
      4.2874      3.6622      3.1160      2.6879      2.4185      2.3277      2.4185      2.6879      3.1160      3.6622
      3.7638      3.0426      2.3747      1.8112      1.4421      1.3228      1.4421      1.8112      2.3747      3.0426
      3.4579      2.6648      1.8841      1.1370      0.5376      0.3932      0.5376      1.1370      1.8841      2.6648
      3.4150      2.6175      1.8292      1.0650      0.4085      0.0802      0.4085      1.0650      1.8292      2.6175
      3.5986      2.8522      2.1412      1.4892      0.9378      0.6669      0.9378      1.4892      2.1412      2.8522
      3.8172      3.0739      2.3477      1.6496      1.0257      0.7042      1.0257      1.6496      2.3477      3.0739
      3.9350      3.1528      2.3716      1.5931      0.8274      0.2577      0.8274      1.5931      2.3716      3.1528
      4.1801      3.4308      2.7078      2.0399      1.5026      1.2741      1.5026      2.0399      2.7078      3.4308
      4.6195      3.9452      3.3274      2.8038      2.4362      2.3007      2.4362      2.8038      3.3274      3.9452

matrix Z':   rows = 10,   cols = 10,   total elements = 100

4.9637      4.2874      3.7638      3.4579      3.4150      3.5986      3.8172      3.9350      4.1801      4.6195
      4.4270      3.6622      3.0426      2.6648      2.6175      2.8522      3.0739      3.1528      3.4308      3.9452
      3.9755      3.1160      2.3747      1.8841      1.8292      2.1412      2.3477      2.3716      2.7078      3.3274
      3.6334      2.6879      1.8112      1.1370      1.0650      1.4892      1.6496      1.5931      2.0399      2.8038
      3.4210      2.4185      1.4421      0.5376      0.4085      0.9378      1.0257      0.8274      1.5026      2.4362
      3.3491      2.3277      1.3228      0.3932      0.0802      0.6669      0.7042      0.2577      1.2741      2.3007
      3.4210      2.4185      1.4421      0.5376      0.4085      0.9378      1.0257      0.8274      1.5026      2.4362
      3.6334      2.6879      1.8112      1.1370      1.0650      1.4892      1.6496      1.5931      2.0399      2.8038
      3.9755      3.1160      2.3747      1.8841      1.8292      2.1412      2.3477      2.3716      2.7078      3.3274
      4.4270      3.6622      3.0426      2.6648      2.6175      2.8522      3.0739      3.1528      3.4308      3.9452
```



### 4.5. Python Helper Functions

```
import sys
import numpy as np
from numpy import linalg as LA
from matplotlib.pyplot import plot
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
from   scipy.linalg import eigvals
from scipy.linalg import toeplitz
import time

def printf(format, *args):
sys.stdout.write(format % args)


def  print1darray(arr, name):
print()
    numberofelements = arr.size
printf("vector %s:    number of elements = %d \n", name, numberofelements);
print()
    for i in range(numberofelements):
printf("%12.4f", arr[i])
    print("\n\n")


def  print2darray(arr2D, name):
print()
    rows, cols = arr2D.shape
    totalelements = arr2D.size
printf("matrix %s:    rows = %d,   cols = %d,   total elements = %d", name, rows, cols, totalelements)
    print("\n")
    for i in range(rows):
        for j in range(cols):
printf("%12.4f", arr2D[i][j])
print()
    print("\n\n")
```

```
def  findaxislimits(xmin, xmax, ymin, ymax):
    if xmin < 0:        axisxmin = xmin * 1.2
    elif xmin == 0:     axisxmin = -1
    else:               axisxmin = xmin / 1.2
    if ymin < 0:        axisymin = ymin * 1.2
    elif ymin == 0:   axisymin = -1
    else:               axisymin = ymin / 1.2
    if xmax < 0:        axisxmax = xmax / 1.2
    elif xmax == 0:   axisxmax = 1
    else:               axisxmax = xmax * 1.2
    if ymax < 0:        axisymax = ymax / 1.2
    elif ymax == 0:   axisymax = 1
    else:               axisymax = ymax * 1.2
    return (axisxmin, axisxmax, axisymin, axisymax)



def  geteigenvalues(A):
    eigenvals = eigvals(A)
    Xi = np.zeros(len(eigenvals))
    Yi = np.zeros(len(eigenvals))
    for i in range(len(eigenvals)):
        Xi[i], Yi[i] = (eigenvals[i].real, eigenvals[i].imag)
    return (Xi, Yi)



def kahan(n, theta=1.2, pert=25):
    s = np.sin(theta)
    c = np.cos(theta)
    u = np.matrix(np.eye(n) - c * np.triu(np.ones((n, n)), 1))
    z = np.matrix(np.diag(s ** np.arange(0, n)))
    eps = np.finfo(float).eps
    u = z * u + pert * eps * np.matrix(np.diag(np.arange(n, 0, -1)))
    return u



def  grcar(n, k=3):
    g = np.tril(np.triu(np.ones((n, n))), k) - np.diag(np.ones(n - 1), -1)
    return g
```

## 4.6. Συνάρτηση gridcalculate

```
def gridcalculate(A, epsilons, N):
    svdtime = 0
    n, cols = A.shape
    norma = LA.norm(A, 2)
## norm(X, 2) returns the 2-norm or maximum singular value of matrix X,
## which is approximately max(svd(X)).
## This value is equivalent to norm(X) in Matlab, but not in Python numpy.

    Ah = (A + A.transpose()) / 2
    Ash = (A - A.transpose()) / 2
    ##  print2darray(Ah, "Ah");
    ##  print2darray(Ash, "Ash");

    ## The eigenvalues of Ah and Ash in ascending order
    EAh, koko =  LA.eig(Ah)    ## LA.eig(Ah) returns a column vector containing
    EAh = sorted(EAh.real)     ## the eigenvalues of square matrix A.
    EAh = np.array(EAh)
    EAsh, koko =  LA.eig(Ash)
    EAsh = sorted(EAsh.imag)
    EAsh = np.array(EAsh)
    ##  print1darray(EAh, "EAh");
    ##  print1darray(EAsh, "EAsh");

    Vxmin = np.zeros((epsilons.size, 1))
    Vxmax = np.zeros((epsilons.size, 1))
    Vymin = np.zeros((epsilons.size, 1))
    Vymax = np.zeros((epsilons.size, 1))
    Vxstep = np.zeros((epsilons.size, 1))
    Vystep = np.zeros((epsilons.size, 1))
    for metr in range(epsilons.size):
        Vxmin[metr] = EAh[0] - epsilons[metr]*norma
        Vxmax[metr] = EAh[-1] + epsilons[metr]*norma
        Vymin[metr] = EAsh[0] - epsilons[metr]*norma
        Vymax[metr] = EAsh[-1] + epsilons[metr]*norma
        Vxstep[metr] = (Vxmax[metr]-Vxmin[metr])/N
        Vystep[metr] = (Vymax[metr]-Vymin[metr])/N

    maxepsilon, indexofmaxepsilon = np.amax(epsilons), np.argmax(epsilons);
    xmin = Vxmin[indexofmaxepsilon]
    xmax = Vxmax[indexofmaxepsilon]
    ymin = Vymin[indexofmaxepsilon]
    ymax = Vymax[indexofmaxepsilon]
    xstep = Vxstep[indexofmaxepsilon]
    ystep = Vystep[indexofmaxepsilon]
```

```
    X = np.zeros((N, 1))
    Y = np.zeros((N, 1))
    Z = np.zeros((N, N))
    for I in range(1, N+1):
        x = xmin + (I-1) * xstep
        X[I-1] = x
        for J in range(1, N+1):
y = ymin + (J-1) * ystep
            Y[J-1] = y
z = complex(x, y)
            currenttime = time.time()
            u, S, vh = np.linalg.svd(z * np.eye(n, dtype=int) - A, full_matrices=True)
            svdtime += time.time() - currenttime
            Z[I-1][J-1] = S[-1]

    return (X, Y, Z, Vxmin, Vxmax, Vymin, Vymax, maxepsilon)
```

## 4.7. Συνάρτηση gridplot

```
def gridplot(X, Y, Z, epsilons, Vxmin, Vxmax, Vymin, Vymax, Xi, Yi,
filename, title, screen_or_file = "screen"):
    mycolour = ['g', 'm', 'b', 'y', 'r', 'c']
    fig, ax = plt.subplots()     ## define Matplotlib figure and axis
    maxepsilon, indexofmaxepsilon = np.amax(epsilons), np.argmax(epsilons);
    xmin = Vxmin[indexofmaxepsilon]
    xmax = Vxmax[indexofmaxepsilon]
    ymin = Vymin[indexofmaxepsilon]
    ymax = Vymax[indexofmaxepsilon]
    axisxmin, axisxmax, axisymin, axisymax = findaxislimits(xmin, xmax, ymin, ymax)
    ax.set_xlim([axisxmin, axisxmax])
    ax.set_ylim([axisymin, axisymax])
plt.title(title)
    for i in range(len(Xi)):
plot(Xi[i], Yi[i], 'k.')
X, Y = np.meshgrid(X, Y)
for i in range(epsilons.size):
ax.contour(X, Y, Z.transpose(), levels=epsilons, colors = ('g', 'm', 'b', 'y', 'r', 'c'))
## contour levels must be increasing
        ax.add_patch(Rectangle((Vxmin[i], Vymin[i]), Vxmax[i]-Vxmin[i], Vymax[i]-Vymin[i],
                               edgecolor = mycolour[i % len(mycolour)], fill=False, lw=0.5))
    if screen_or_file == "file" :
fig.savefig(filename, dpi=1200)
    elif screen_or_file == "screen" :
plt.show()
plt.close('all')
```

Με την τελευταία παράμετρο, επιλέγουμε αν θέλουμε τα γραφήματα να εμφανιστούν στην οθόνη ή να αποθηκευτούν σε αρχεία. Τα ονόματα των αρχείων δημιουργούνται αυτόματα.

```
gridplot(X, Y, Z, epsilons, Vxmin, Vxmax, Vymin, Vymax, Xi, Yi, filename, title, "screen")

gridplot(X, Y, Z, epsilons, Vxmin, Vxmax, Vymin, Vymax, Xi, Yi, filename, title, "file")
```

## 4.8. Python check_pseudospectra σε Πίνακες Τυχαίων Αριθμών

Θα κατασκευάσουμε πίνακες τυχαίων αριθμών για διαστάσεις **dim** από 5x5 ως 100x100.

Θα κρατήσουμε σταθερή την ακρίβεια **N**(πλήθος σημείων του πλέγματος ανά άξονα) στο 50.

Θα υπολογίσουμε και θα απεικονίσουμε σε γράφημα το ψευδοφάσμα του A
για διαφορετικές τιμές του ε (0.05, 0.2, 0.5, 0.8, 1), **epsilons**,
καθώς και το χωρίο Ω που κατασκεύασε ο αλγόριθμος για κάθε περίπτωση.

Συμπεράσματα:

- Σε όλες τις περιπτώσεις το χωρίο Ω περιλάμβανε ολόκληρο το ψευδοφάσμα (ακόμη και στα γραφήματα που έχουμε κάνει μεγέθυνση και δε φαίνεται).
- Ο υπολογισμός του ψευδοφάσματος χρειάζεται μεγαλύτερη ακρίβεια όσο μεγαλώνει η διάσταση, και όσο μικραίνει το ε.

## Κώδικας συνάρτησης check_pseudospectra

```
defcheck_pseudospectra(algorithm = "grid", data = "random"):
if (algorithm == "grid" or algorithm == "inex"):
        epsilons = np.array([0.05, 0.2, 0.5, 0.8, 1])
        N_array = [50]
    elif (algorithm == "kahan" or algorithm == "grcar"):
        N_array = [400]
        epsilons = np.array([0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02])
    dimensions = [5, 10, 25, 50, 100, 200]
    for dim in dimensions:
        if data == "random":
            A = np.random.uniform(low=0.0, high=1.0, size=(dim, dim))
        elif data == "kahan":
            A = kahan(dim)
        elif data == "grcar":
            A = grcar(dim)
        Xi, Yi = geteigenvalues(A)
        for N in N_array:
                currenttime = time.time()
                if algorithm == "grid":
X, Y, Z, Vxmin, Vxmax, Vymin, Vymax, maxepsilon, svdtime = gridcalculate(A, epsilons, N)
print("algorithm: grid, dim = %d, N = %d, time = %.6f seconds, svdtime = %.6f seconds"
                                % (dim, N, time.time() - currenttime, svdtime), flush=True)
                    filename = "pse_grid_" + "dim_" + str(dim) + "_N_" + str(N) + ".jpg"
                    title = "pse_grid_" + "dim_" + str(dim) + "_N_" + str(N)
gridplot(X, Y, Z, epsilons, Vxmin, Vxmax, Vymin, Vymax, Xi, Yi, filename, title, "screen")
                elif algorithm == "inex":
X, Y, Z, Vxmin, Vxmax, Vymin, Vymax, maxepsilon, svdtime, svdcount, exclusiontime =
inexcalculate(A, epsilons, N)
print("algorithm: inex, dim = %d, N = %d, time = %.6f seconds, svdtime = %.6f seconds,
svdcount = %d out of %d, exclusiontime = %.6f seconds"
                                    % (dim, N, time.time() - currenttime, svdtime, svdcount, N*N, exclusiontime), flush=True)
                    filename = "pse_inex_" + "dim_" + str(dim) + "_N_" + str(N) + ".jpg"
                    title = "pse_inex_" + "dim_" + str(dim) + "_N_" + str(N)
gridplot(X, Y, Z, epsilons, Vxmin, Vxmax, Vymin, Vymax, Xi, Yi, filename, title, "screen")
```

## Κλήση συνάρτησης

```
check_pseudospectra(algorithm = "grid", data = "random")
```

## Output

```
algorithm: grid, dim = 5, N = 50, time = 0.201646 seconds, svdtime = 0.178129 seconds
algorithm: grid, dim = 10, N = 50, time = 0.365266 seconds, svdtime = 0.330736 seconds
algorithm: grid, dim = 25, N = 50, time = 1.591566 seconds, svdtime = 1.542037 seconds
algorithm: grid, dim = 50, N = 50, time = 4.389621 seconds, svdtime = 4.344524 seconds
algorithm: grid, dim = 100, N = 50, time = 33.778812 seconds, svdtime = 33.666790 seconds
algorithm: grid, dim = 200, N = 50, time = 162.277393 seconds, svdtime = 161.961666 seconds
```

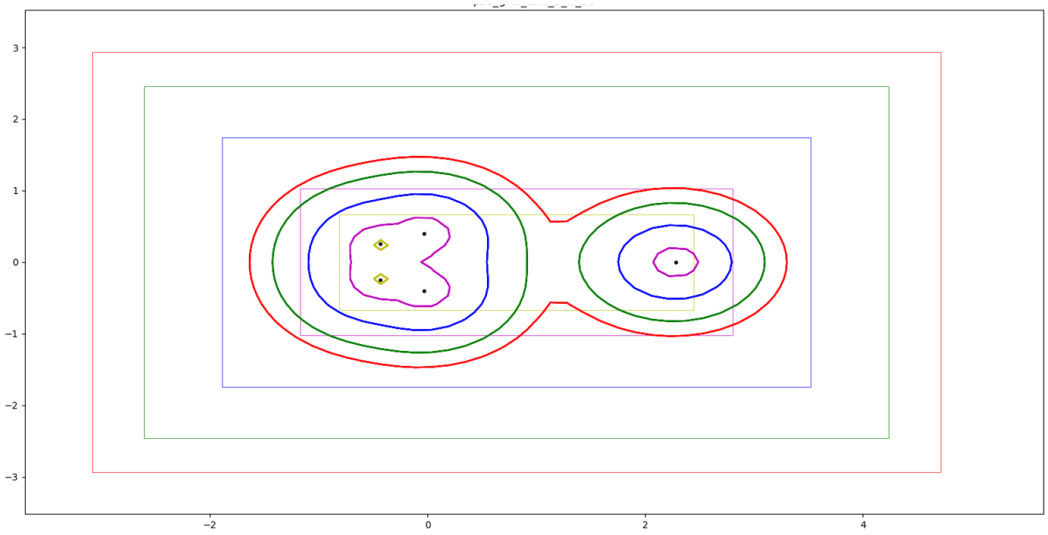
Μετά από κάθε γραμμή εμφανίζεται στην οθόνη το αντίστοιχο γράφημα

N = 50 για όλα τα γραφήματα

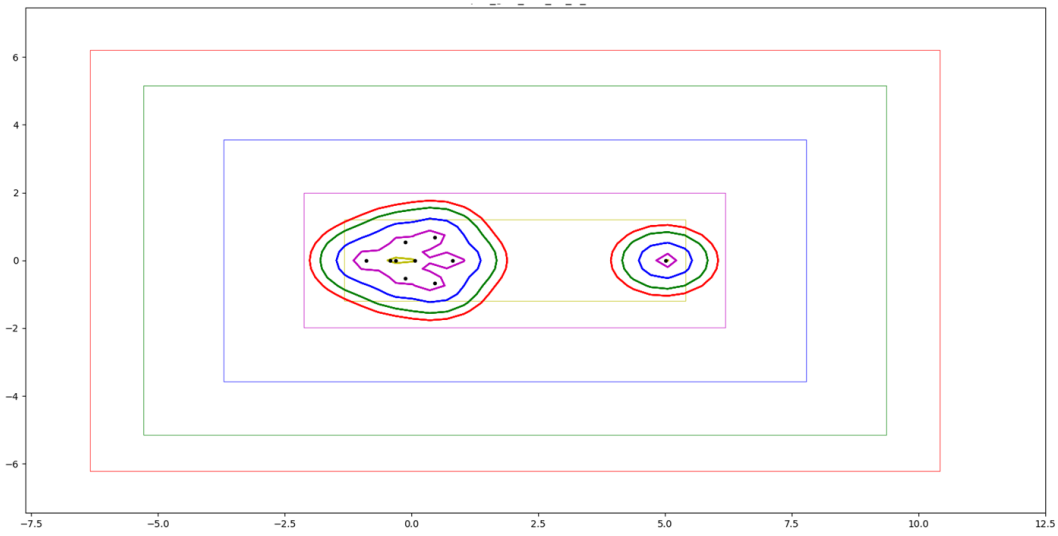
κόκκινο: ε = 1, πράσινο: ε = 0.8, μπλε: ε = 0.5,

ματζέντα: ε = 0.2, κίτρινο: ε = 0.05

dim = 5




dim = 10

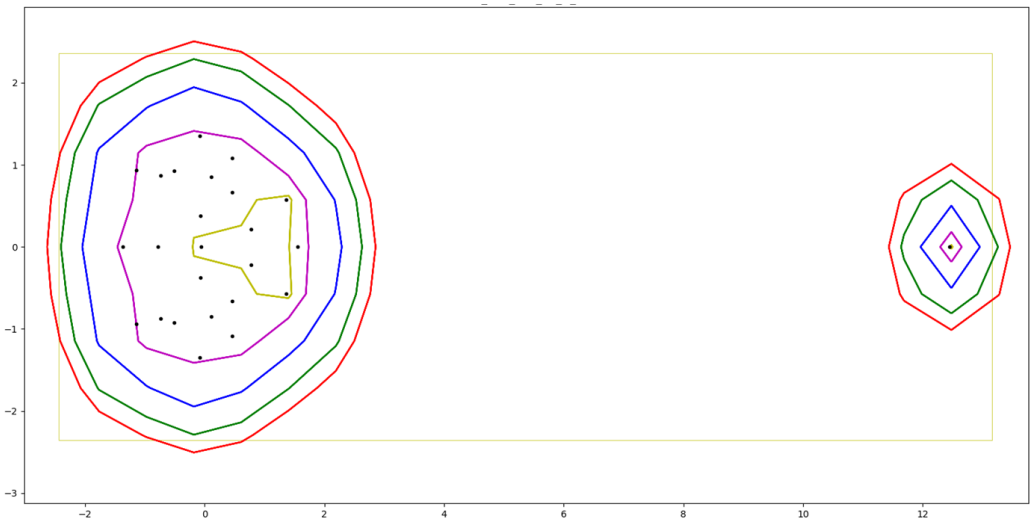


N = 50 για όλα τα γραφήματα

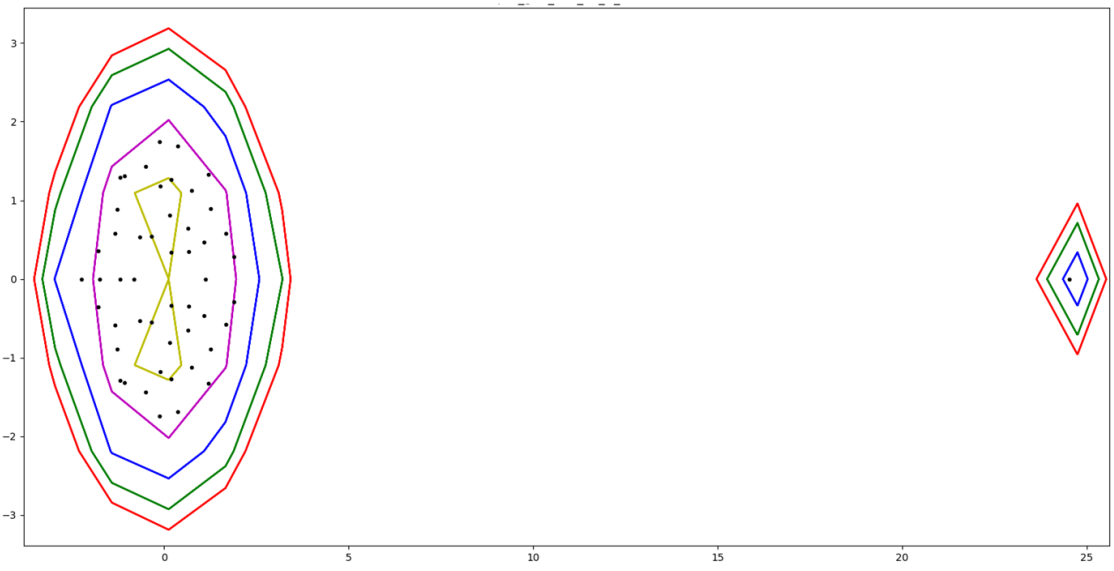
κόκκινο: ε = 1, πράσινο: ε = 0.8, μπλε: ε = 0.5,

ματζέντα: ε = 0.2, κίτρινο: ε = 0.05

dim = 25



dim = 50

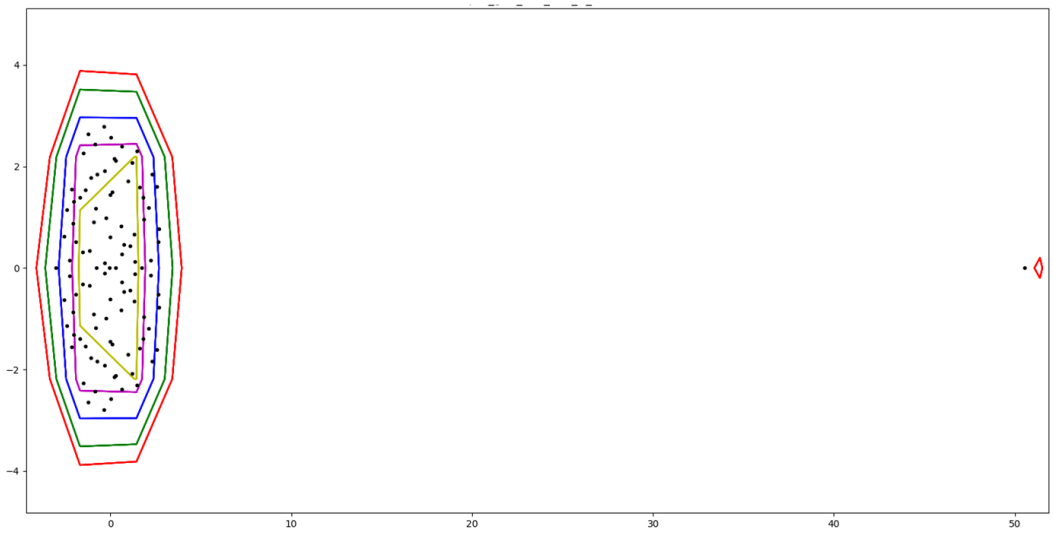


N = 50 για όλα τα γραφήματα

κόκκινο: ε = 1, πράσινο: ε = 0.8, μπλε: ε = 0.5,

ματζέντα: ε = 0.2, κίτρινο: ε = 0.05

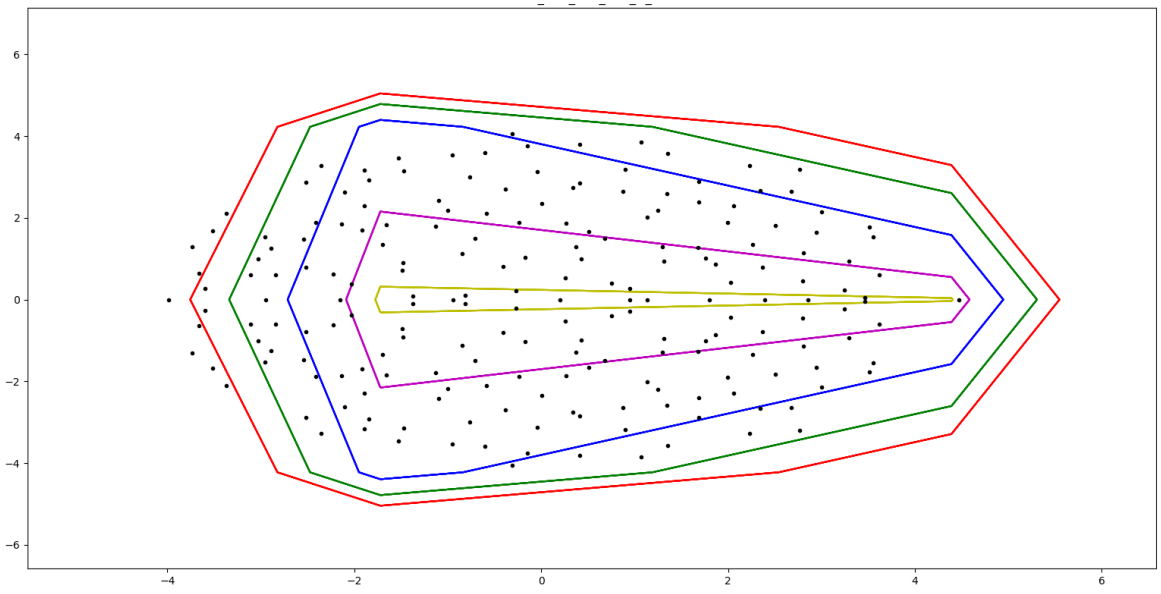
dim = 100




dim = 200

λεπτομέρεια

η μακρινή δεξιά ιδιοτιμή δεν μπήκε στο ψευδοφάσμα



## 4.9. Κώδικας Python για Έλεγχο της Επίδρασης του ε στον Χρόνο

**Το μέγεθος του προβλήματος δεν εξαρτάται από την επιλογή του epsilon** όπως φαίνεται από το ακόλουθο πείραμα. Είναι λογικό, καθώς έχουμε πάνω κάτω τους ίδιους υπολογισμούς, απλώς αυτοί γίνονται σε μεγαλύτερο γεωμετρικά χωρίο. Το γεγονός ότι οι υπολογιζόμενοι αριθμοί είναι μεγαλύτεροι αριθμητικά δεν επηρεάζει σε κάτι τον χρόνο υπολογισμού.

Κώδικας

```
N_array = [20]
extension = [0.0001, 0.01, 1, 100]
epsilon_array = [np.array([0.0001]), np.array([0.01]), np.array([1]), np.array([100])]
dimensions = [5, 10, 25, 50, 100, 200]
for dim in dimensions:
    A = np.random.uniform(low=0.0, high=1.0, size=(dim, dim))
    Xi, Yi = geteigenvalues(A)
    for N in N_array:
         for i in range(len(epsilon_array)):
            epsilons = epsilon_array[i]
            currenttime = time.time()
X,Y,Z, Vxmin, Vxmax, Vymin, Vymax, maxepsilon, svdtime = gridcalculate(A, epsilons, N)
print("algorithm: grid, dim = %d, N = %d, epsilon = %.2f, time = %.6f seconds "
                 % (dim, N, epsilons, time.time() - currenttime), flush=True)
```

Program Output

```
algorithm: grid, dim = 5, N = 20, epsilon = 0.00, time = 0.036024 seconds
algorithm: grid, dim = 5, N = 20, epsilon = 0.01, time = 0.034522 seconds
algorithm: grid, dim = 5, N = 20, epsilon = 1.00, time = 0.034525 seconds
algorithm: grid, dim = 5, N = 20, epsilon = 100.00, time = 0.035022 seconds
algorithm: grid, dim = 10, N = 20, epsilon = 0.00, time = 0.055037 seconds
algorithm: grid, dim = 10, N = 20, epsilon = 0.01, time = 0.055537 seconds
algorithm: grid, dim = 10, N = 20, epsilon = 1.00, time = 0.057037 seconds
algorithm: grid, dim = 10, N = 20, epsilon = 100.00, time = 0.058040 seconds
algorithm: grid, dim = 25, N = 20, epsilon = 0.00, time = 0.224238 seconds
algorithm: grid, dim = 25, N = 20, epsilon = 0.01, time = 0.224736 seconds
algorithm: grid, dim = 25, N = 20, epsilon = 1.00, time = 0.235772 seconds
algorithm: grid, dim = 25, N = 20, epsilon = 100.00, time = 0.228679 seconds
algorithm: grid, dim = 50, N = 20, epsilon = 0.00, time = 0.731869 seconds
algorithm: grid, dim = 50, N = 20, epsilon = 0.01, time = 0.728039 seconds
algorithm: grid, dim = 50, N = 20, epsilon = 1.00, time = 0.739521 seconds
algorithm: grid, dim = 50, N = 20, epsilon = 100.00, time = 0.727508 seconds
algorithm: grid, dim = 100, N = 20, epsilon = 0.00, time = 5.473539 seconds
algorithm: grid, dim = 100, N = 20, epsilon = 0.01, time = 5.491022 seconds
algorithm: grid, dim = 100, N = 20, epsilon = 1.00, time = 6.611641 seconds
algorithm: grid, dim = 100, N = 20, epsilon = 100.00, time = 6.694494 seconds
algorithm: grid, dim = 200, N = 20, epsilon = 0.00, time = 26.462770 seconds
algorithm: grid, dim = 200, N = 20, epsilon = 0.01, time = 26.459198 seconds
algorithm: grid, dim = 200, N = 20, epsilon = 1.00, time = 26.337740 seconds
algorithm: grid, dim = 200, N = 20, epsilon = 100.00, time = 25.944083 seconds
```

## 4.10. Συνάρτηση check_time σε Πίνακες Τυχαίων Αριθμών για Χρονομέτρηση

Θα κατασκευάσουμε πίνακες τυχαίων αριθμών για διαστάσεις από 2x2 ως 200 x200.

Θα κρατήσουμε σταθερό το ε = 1.

Θα υπολογίσουμε το ψευδοφάσμα του *A* για διαφορετικές τιμές της ακρίβειας,από N = 5 ως N = 200.

Συμπεράσματα:

Όσον αφορά τον χρόνο εκτέλεσης του αλγόριθμου Grid, συμπεραίνουμε ότι ο **χρόνος εκτέλεσης είναι γραμμικός σε σχέση με το μέγεθος του προβλήματος**. Μικροδιαφορές υπάρχουν, αλλά οφείλονται πιθανότατα σε ζητήματα του hardware των υπολογιστών, και πιο συγκεκριμένα στο φαινόμενο του caching (τοποθέτηση δεδομένων στην κρυφή μνήμη του επεξεργαστή).

**Το μέγεθος του προβλήματος εξαρτάται από δύο παράγοντες:**

- **τη διάσταση dim του δισδιάστατου πίνακα**(dimxdim) που μελετάμε
- **την ακρίβεια,** δηλαδή τον αριθμό N των σημείων που μελετάμε κατά xκαι κατά y (NxN συνολικά)

**Το μέγεθος του προβλήματος δεν εξαρτάται από την επιλογή του epsilon** όπως είδαμε σε προηγούμενη ενότητα.

Τονίζουμε ότι **πρόκειται για τετραγωνικούς πίνακες**, οπότε όταν περνάμε, για παράδειγμα, από διάσταση 50 σε διάσταση 100, το μέγεθος του προβλήματος τετραπλασιάζεται, καθώς ο πίνακας από 50 x 50 γίνεται 100 x 100. Το ίδιο ισχύει και με την ακρίβεια, καθώς το grid σημείων που παίρνουμε είναι δισδιάστατο, κατά xκαι κατά y.

**Σχεδόν όλος ο χρόνος αφορά τις svd παραγοντοποιήσεις.**

Κώδικας

```
def check_time(algorithm = "grid", data = "random"):
    epsilons = np.array([1])       ## epsilon = 1, constant
    N_array = [5, 10, 20, 50, 100, 200]
    dimensions = [2, 5, 10, 25, 50, 100, 200]
    if data == "random" :
print("Data: matrix of random numbers \n")
    elif data == "kahan" :
print("Data: kahan matrix \n")
    elif data == "grcar" :
print("Data: grcar matrix \n")
    for dim in dimensions:
        if data == "random" :
            A = np.random.uniform(low=0.0, high=1.0, size=(dim, dim))
        elif data == "kahan" :
            A = kahan(dim)
        elif data == "grcar" :
            A = grcar(dim)
        Xi, Yi = geteigenvalues(A)
        for N in N_array:
                currenttime = time.time()
                if algorithm == "grid":
                    X, Y, Z, Vxmin, Vxmax, Vymin, Vymax, maxepsilon, svdtime =
gridcalculate(A, epsilons, N)
print("algorithm: grid, dim = %d, N = %d, time = %.6f seconds, svdtime = %.6f seconds"
                          % (dim, N, time.time() - currenttime, svdtime), flush=True)
                elif algorithm == "inex":
X, Y, Z, Vxmin, Vxmax, Vymin, Vymax, maxepsilon, svdtime, svdcount, exclusiontime =
inexcalculate(A, epsilons, N)
print("algorithm: inex, dim = %d, N = %d, time = %.6f seconds,
svdtime = %.6f seconds, svdcount = %d out of %d, exclusiontime = %.6f seconds"
                           % (dim, N, time.time() - currenttime, svdtime, svdcount,
N*N, exclusiontime), flush=True)
```

Κλήση συνάρτησης check_time

```
check_time("grid", "random")
```

Αποτελέσματα χρονομέτρησης αλγορίθμου grid σε πίνακες τυχαίων αριθμών

| dim | N | time | svdtime |
|---|---|---|---|
| 2 | 5 | 0.00 | 0.00 |
| 2 | 10 | 0.01 | 0.01 |
| 2 | 20 | 0.04 | 0.03 |
| 2 | 50 | 0.24 | 0.19 |
| 2 | 100 | 0.84 | 0.66 |
| 2 | 200 | 4.87 | 3.87 |
| 5 | 5 | 0.00 | 0.00 |
| 5 | 10 | 0.01 | 0.01 |
| 5 | 20 | 0.05 | 0.05 |
| 5 | 50 | 0.27 | 0.22 |
| 5 | 100 | 1.08 | 0.89 |
| 5 | 200 | 5.56 | 4.64 |
| 10 | 5 | 0.01 | 0.01 |
| 10 | 10 | 0.02 | 0.02 |
| 10 | 20 | 0.08 | 0.07 |
| 10 | 50 | 0.47 | 0.42 |
| 10 | 100 | 1.95 | 1.76 |
| 10 | 200 | 10.08 | 9.06 |
| 25 | 5 | 0.03 | 0.02 |
| 25 | 10 | 0.12 | 0.11 |
| 25 | 20 | 0.38 | 0.36 |
| 25 | 50 | 2.20 | 2.11 |
| 25 | 100 | 8.37 | 8.03 |
| 25 | 200 | 46.51 | 45.20 |
| 50 | 5 | 0.07 | 0.06 |
| 50 | 10 | 0.26 | 0.25 |
| 50 | 20 | 1.01 | 0.99 |
| 50 | 50 | 6.57 | 6.46 |
| 50 | 100 | 25.42 | 25.02 |
| 50 | 200 | 129.56 | 127.92 |
| 100 | 5 | 0.45 | 0.41 |
| 100 | 10 | 1.68 | 1.65 |
| 100 | 20 | 6.59 | 6.54 |
| 100 | 50 | 41.51 | 41.37 |
| 100 | 100 | 201.97 | 201.35 |
| 100 | 200 | 981.37 | 978.90 |
| 200 | 5 | 2.40 | 2.15 |
| 200 | 10 | 8.77 | 8.51 |
| 200 | 20 | 33.70 | 33.42 |
| 200 | 50 | 208.51 | 208.13 |
| 200 | 100 | 856.82 | 855.96 |
| 200 | 200 | 4143.22 | 4138.74 |

## 4.11. Συνάρτηση check_precision σε Πίνακες Τυχαίων Αριθμών για Εύρεση Ικανής Ακρίβειας

Επειδή **η διάσταση ενός πίνακα είναι δεδομένη** (από το πρόβλημα που έχουμε να λύσουμε), έχει σημασία **να βρούμε την ελάχιστη ακρίβεια η οποία είναι ικανοποιητική**, μας δίνει δηλαδή μια καλή εικόνα του ψευδοφάσματος, συναρτήσει πάντα της διάστασης του πίνακα.

Δοκιμάζουμε ε=0.01 και ε = 1, για διαστάσεις από 5 x 5 έως και 200 x 200, και ακρίβειες από 5 x 5 έως και 200 x 200.

Από τα σχήματα των επόμενων σελίδων, συμπεραίνουμε τα εξής (για πίνακες τυχαίων αριθμών):

- Για ε = 1, ακρίβεια ίση με τη διάσταση δίνει γενικά άριστα αποτελέσματα.
- Για ε = 1, για διαστάσεις έως και 200 x 200, ακρίβεια 50 x 50 είναι σχεδόν άψογη,
- Για ε = 1, για διαστάσεις έως και 50 x 50, ακρίβεια 20 x 20 είναι ικανοποιητική.
- Για ε =0.01, έως και 10 x 10, ακρίβεια 200 x 200 δίνει άριστα αποτελέσματα.
- Για ε =0.01, για διαστάσεις πάνω από 10 x 10, χρειάζεται ακρίβεια πάνω από 200 x 200

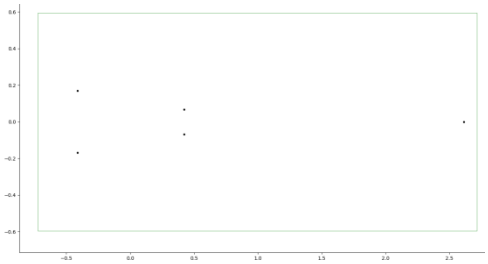
Κλήση: `check_precision("grid", "random")`

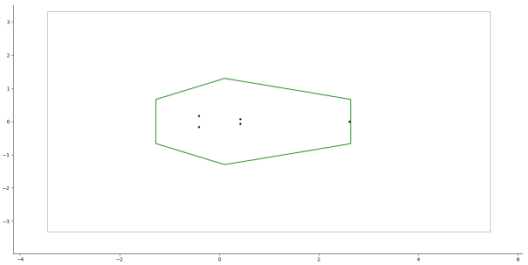
## Κώδικας συνάρτησης check_precision

```
def check_precision(algorithm = "grid", data = "random"):

    ###  Code to find a good enough precision, varying epsilon, precision, and dimension
    if data == "random" :
print("Data: matrix of random numbers \n")
    N_array = [5, 10, 20, 50, 100, 200]
    epsilon_array = [np.array([0.01]), np.array([1])]
    extension = [0.01, 1]
    dimensions = [5, 10, 25, 50, 100, 200]
    for dim in dimensions:
        if data == "random" :
            A = np.random.uniform(low=0.0, high=1.0, size=(dim, dim))
        Xi, Yi = geteigenvalues(A)
        for N in N_array:
             for i in range(len(epsilon_array)):
                epsilons = epsilon_array[i]
                currenttime = time.time()

                if algorithm == "grid":
                    X, Y, Z, Vxmin, Vxmax, Vymin, Vymax, maxepsilon, svdtime = gridcalculate(A, epsilons, N)
print("algorithm: grid, dim = %d, N = %d, epsilon = %s, time = %.6f seconds"
                                % (dim, N, str(extension[i]), time.time() - currenttime), flush=True)
                    filename = "pse_grid_" + "dim_" + str(dim) + "_N_" + str(N) + "_eps_" + str(extension[i]) + ".jpg"
                    title = "pse_grid_" + "dim_" + str(dim) + "_N_" + str(N) + "_eps_" + str(extension[i])
gridplot(X, Y, Z, epsilons, Vxmin, Vxmax, Vymin, Vymax, Xi, Yi, filename, title, "screen")

                elif algorithm == "inex":
X, Y, Z, Vxmin, Vxmax, Vymin, Vymax, maxepsilon, svdtime, svdcount, exclusiontime
= inexcalculate(A, epsilons, N)
print("algorithm: inex, dim = %d, N = %d, epsilon = %s, time = %.6f seconds"
                                % (dim, N, str(extension[i]), time.time() - currenttime), flush=True)
                    filename = "pse_inex_" + "dim_" + str(dim) + "_N_" + str(N) + ".jpg"
                    title = "pse_inex_" + "dim_" + str(dim) + "_N_" + str(N)
gridplot(X, Y, Z, epsilons, Vxmin, Vxmax, Vymin, Vymax, Xi, Yi, filename, title, "screen")
```

ε = 0.01 ε = 1

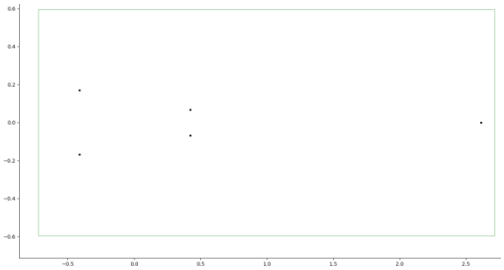
dim = 5, N = 5

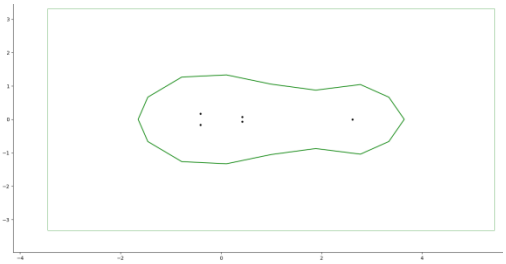




ε = 0.01 ε = 1

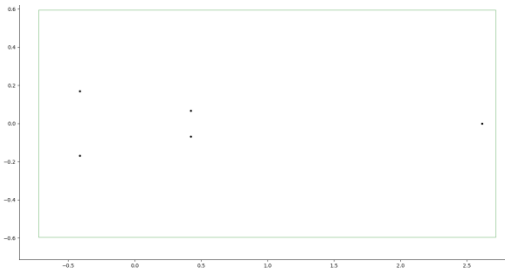
dim = 5, N = 10

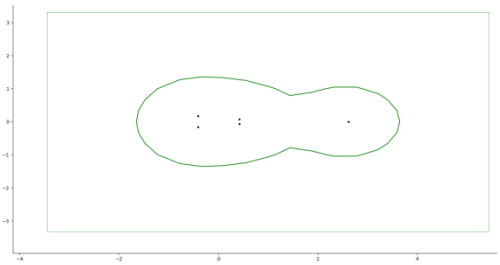




ε = 0.01 ε = 1

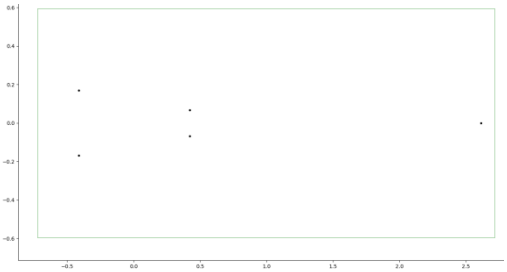
dim = 5, N = 20

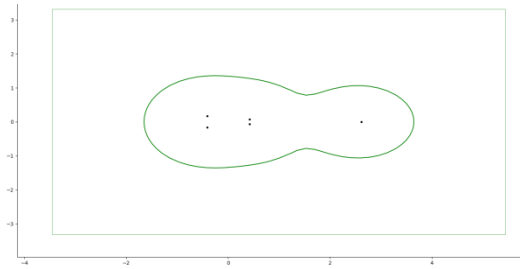




ε = 0.01 ε = 1

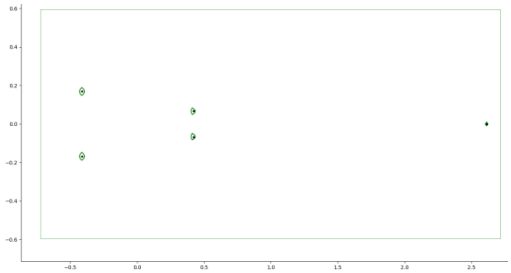
dim = 5, N = 50

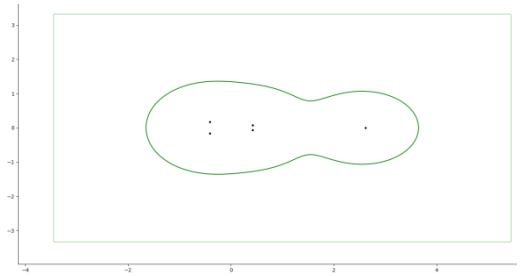




ε = 0.01 ε = 1

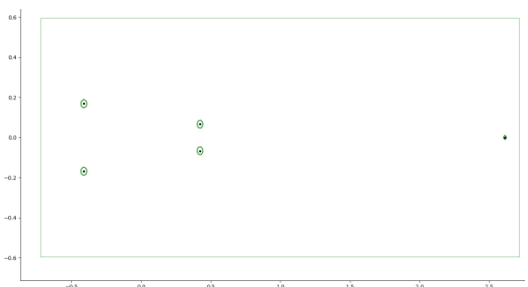
dim = 5, N = 100

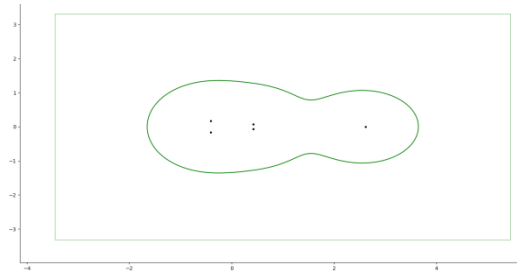




ε = 0.01 ε = 1

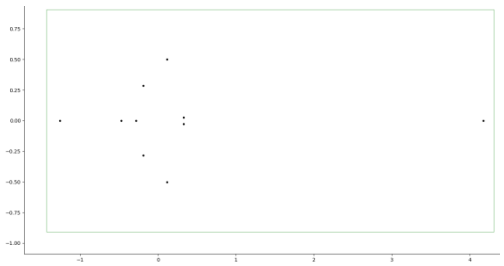
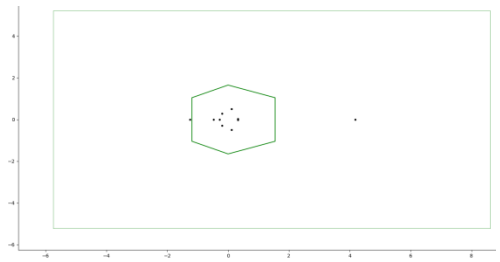
dim = 5, N = 200





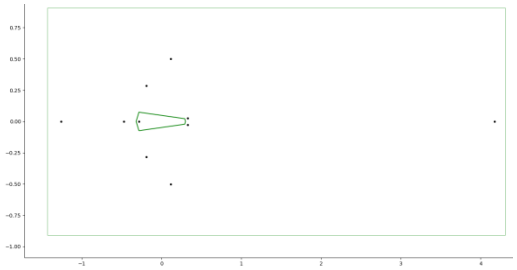
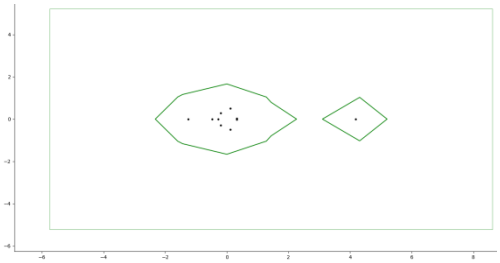
ε = 0.01 ε = 1

dim = 10, N = 5

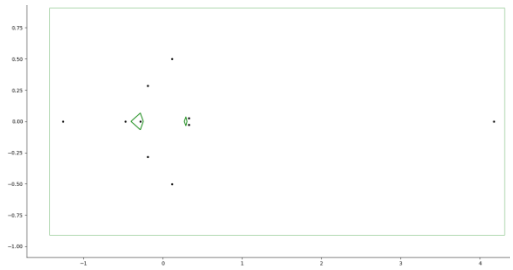
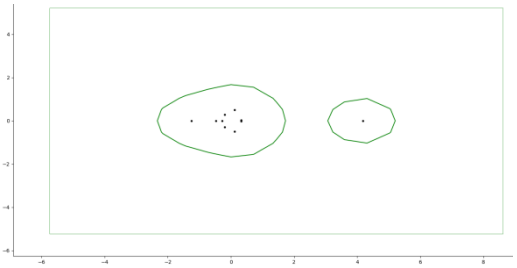
ε = 0.01 ε = 1

dim = 10, N = 10

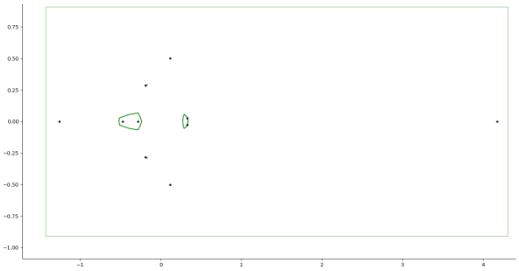
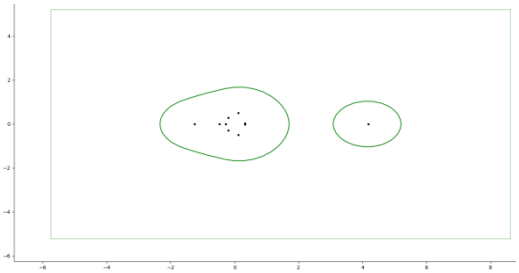
ε = 0.01 ε = 1

dim = 10, N = 20

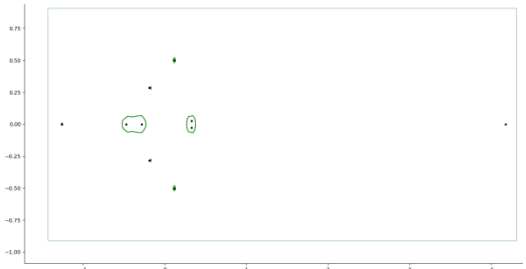
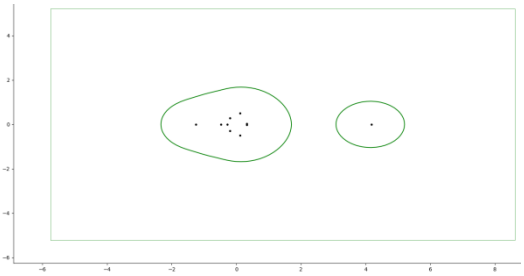
ε = 0.01 ε = 1

dim = 10, N = 50

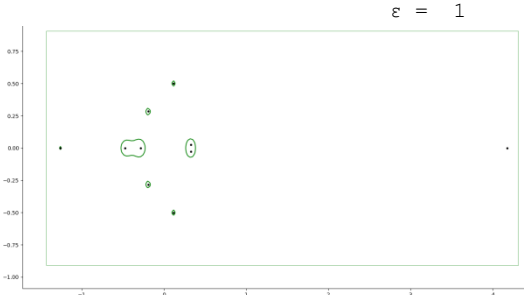
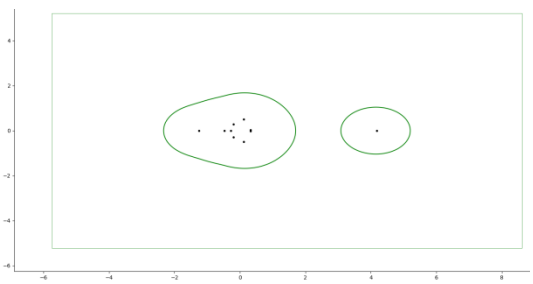
ε = 0.01 ε = 1

dim = 10, N = 100

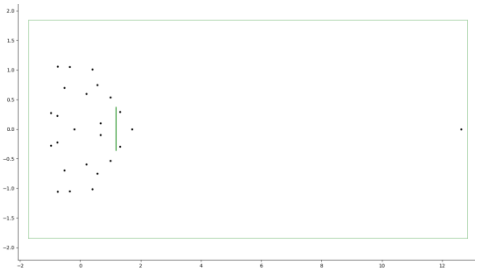
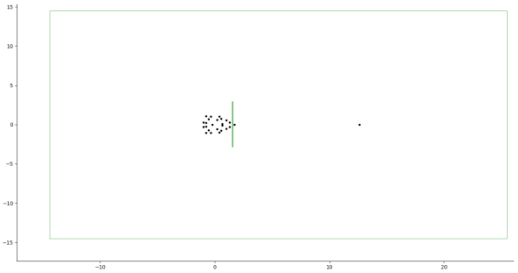
ε = 0.01 ε = 1

dim = 10, N = 200

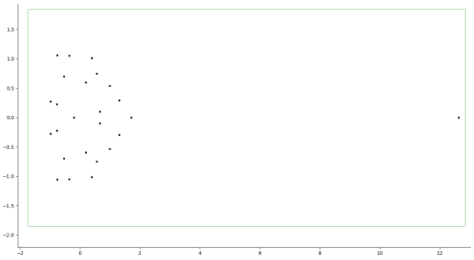
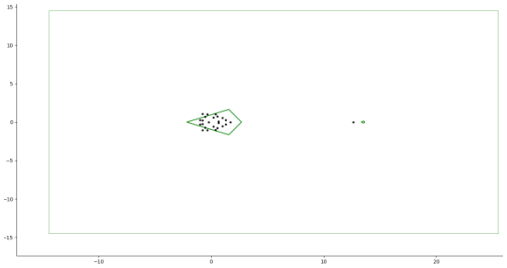
ε = 0.01 ε = 1

dim = 25, N = 5

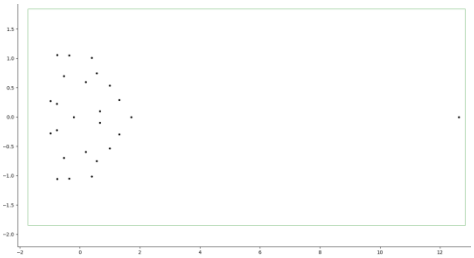
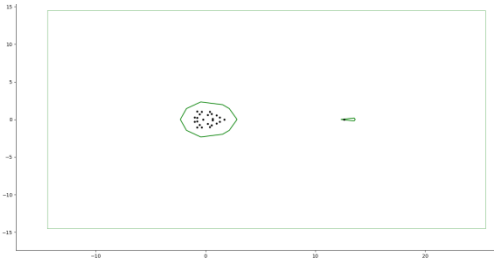
ε = 0.01 ε = 1

dim = 25, N = 10

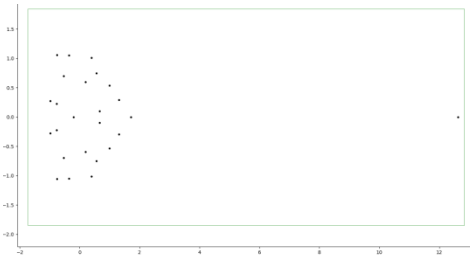
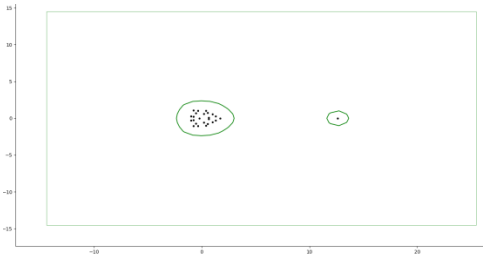
ε = 0.01 ε = 1

dim = 25, N = 20

ε = 0.01 ε = 1

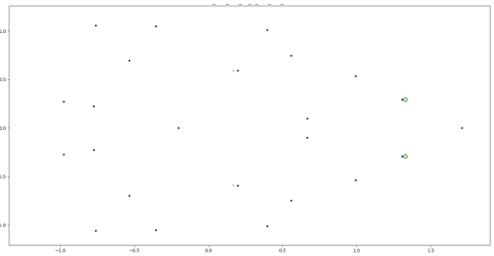
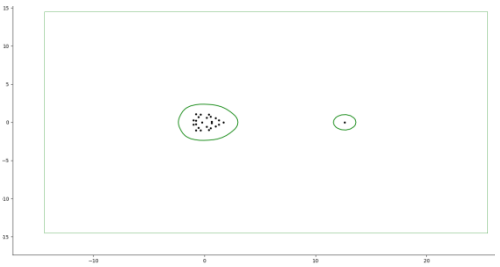
dim = 25, N = 50

ε = 0.01 ε = 1

dim = 25, N = 100

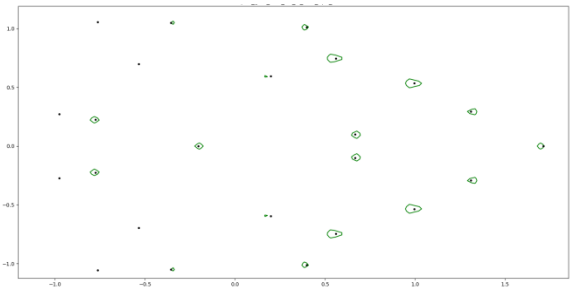
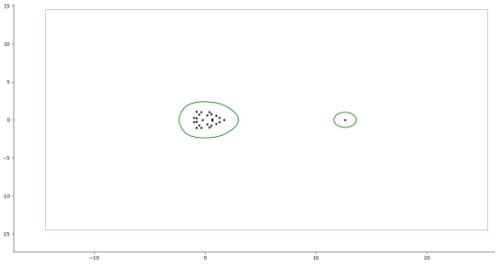
Λεπτομέρεια (ε = 0.01)

ε = 0.01 ε = 1

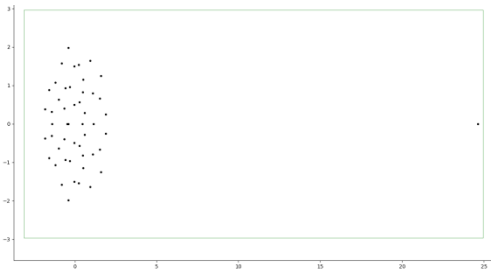
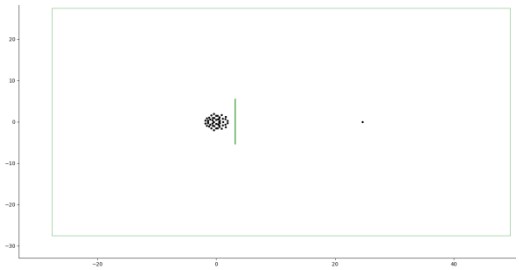
dim = 25, N = 200

Λεπτομέρεια (ε = 0.01)

ε = 0.01 ε = 1

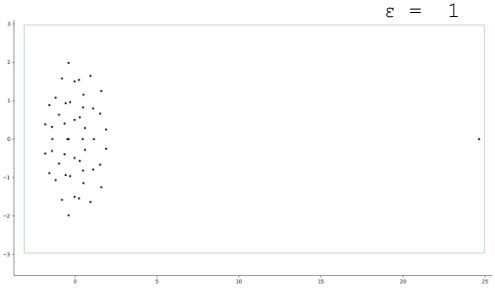
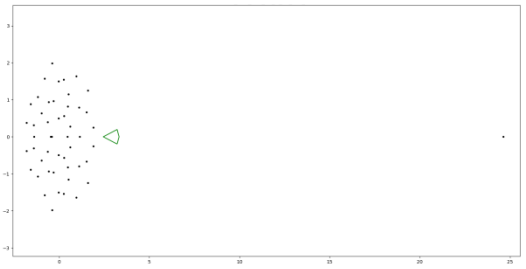
dim = 50, N = 5

ε = 0.01 ε = 1

dim = 50, N = 10

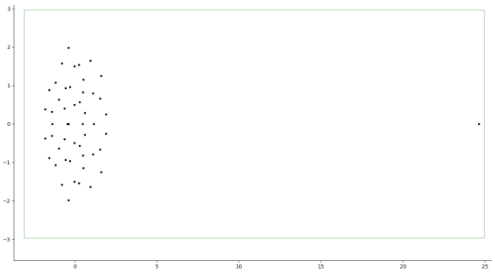
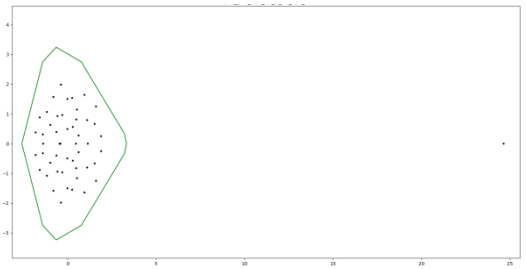
λεπτομέρεια (ε = 1)

ε = 0.01 ε = 1

dim = 50, N = 20

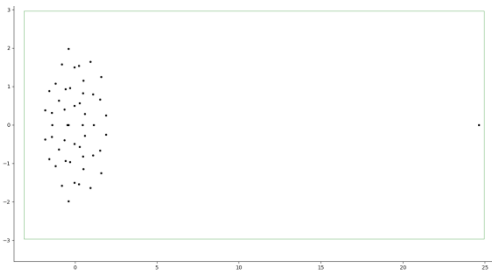
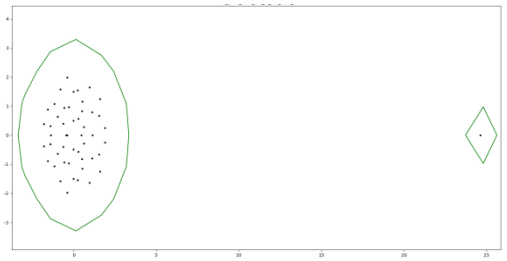
λεπτομέρεια (ε = 1)

ε = 0.01 ε = 1

dim = 50, N = 50

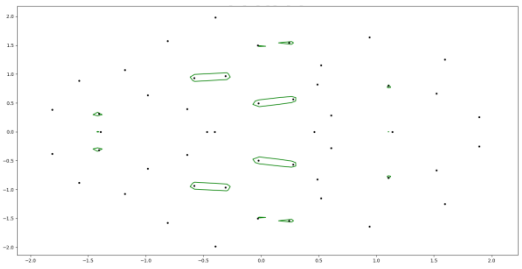
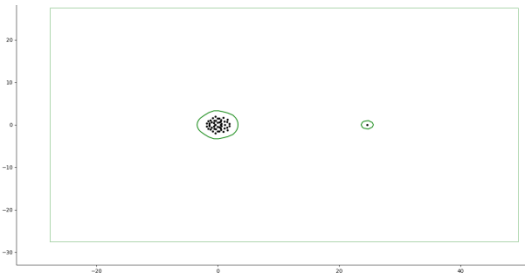
λεπτομέρεια (ε = 1)

ε = 0.01 ε = 1

dim = 50, N = 100

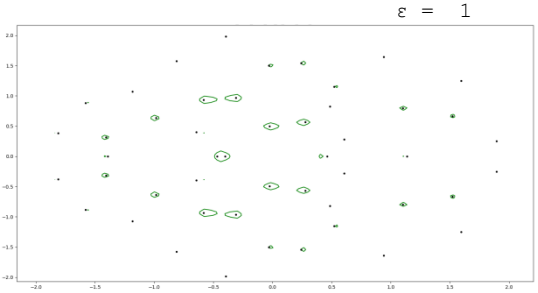
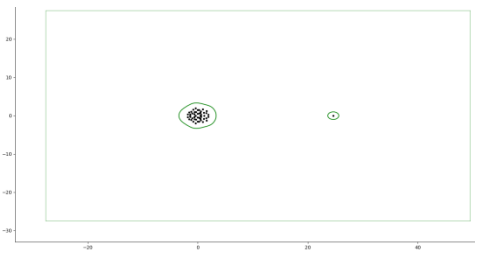
λεπτομέρεια (ε = 0.01)

ε = 0.01 ε = 1

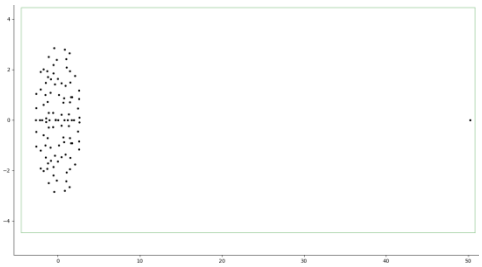
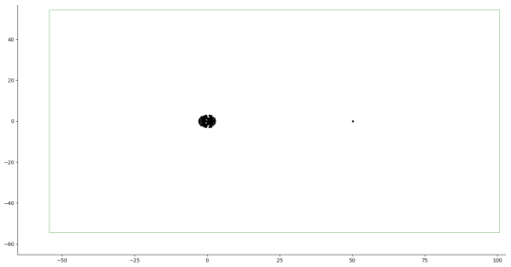
dim = 50, N = 200

λεπτομέρεια (ε = 0.01)

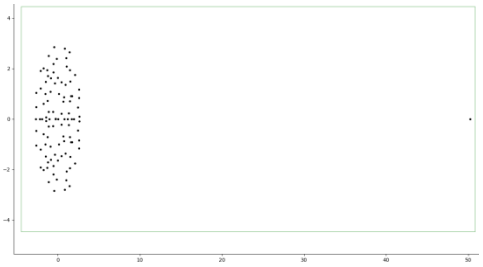
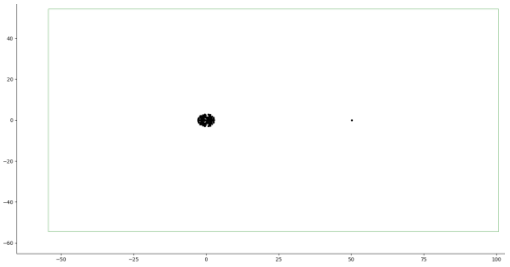
ε = 0.01 ε = 1

dim = 100, N = 5

ε = 0.01 ε = 1

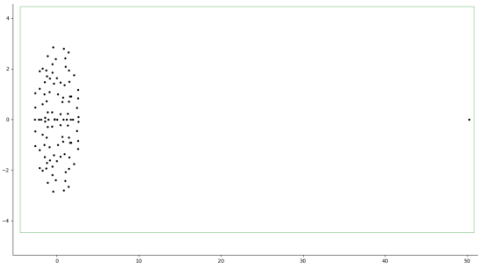
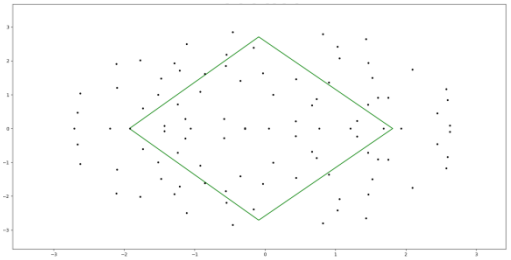
dim = 100, N = 10

ε = 0.01 ε = 1

dim = 100, N = 20

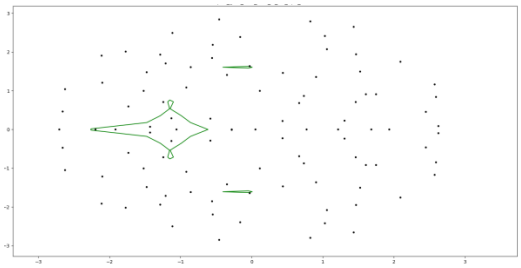
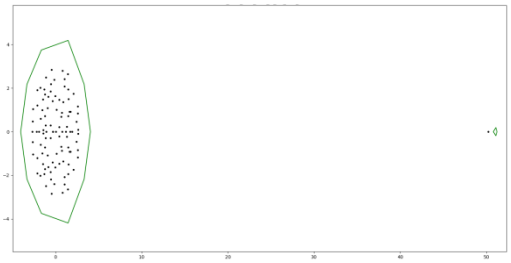
λεπτομέρεια (ε = 1)

ε = 0.01 ε = 1

dim = 100, N = 50

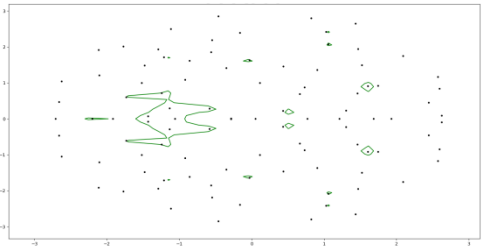
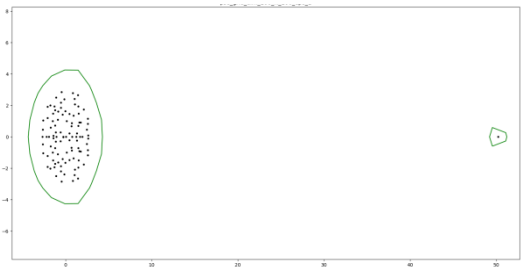
λεπτομέρεια (ε = 0.01, 0.1)

ε = 0.01 ε = 1

dim = 100, N = 100

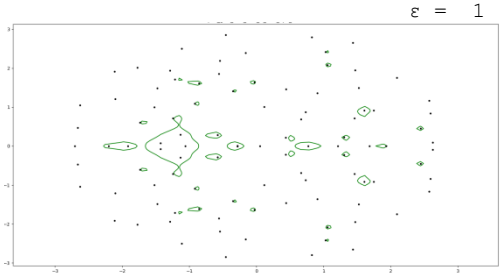
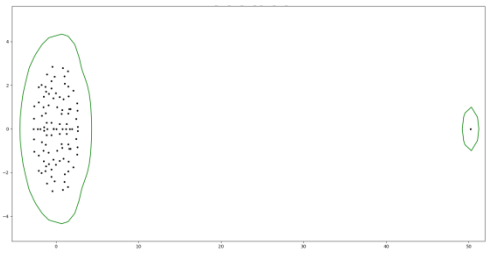
λεπτομέρεια (ε = 0.01, 0.1)

ε = 0.01 ε = 1

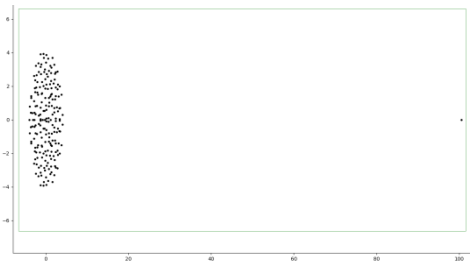
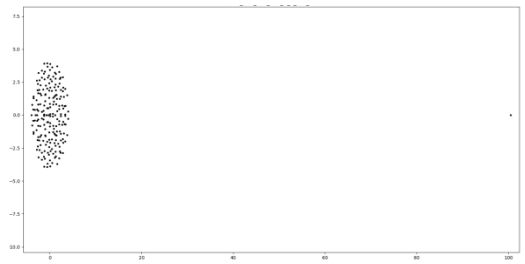
dim = 100, N = 200

λεπτομέρεια (ε = 0.01, 0.1)

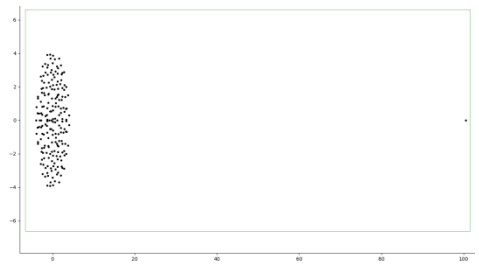
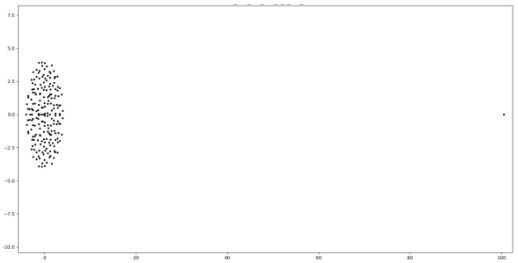
ε = 0.01 ε = 1

dim = 200, N = 5

ε = 0.01 ε = 1

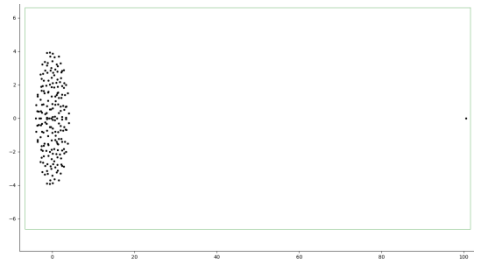
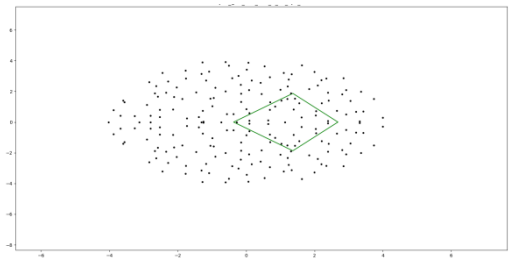
dim = 200, N = 10

ε = 0.01 ε = 1

dim = 200, N = 20

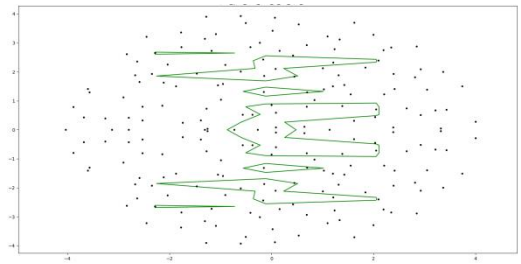
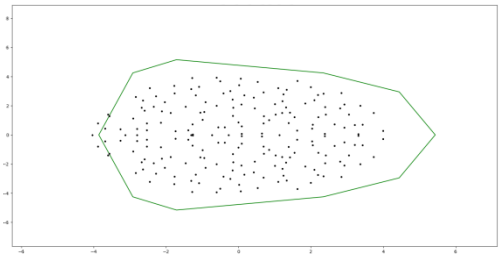
λεπτομέρεια (ε = 1)

ε = 0.01 ε = 1

dim = 200, N = 50

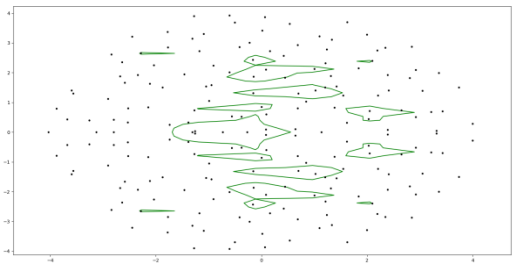
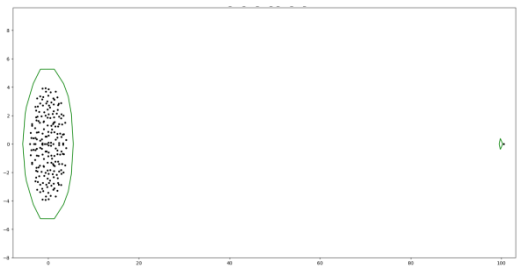
λεπτομέρεια (ε = 0.01, 0.1)

ε = 0.01 ε = 1

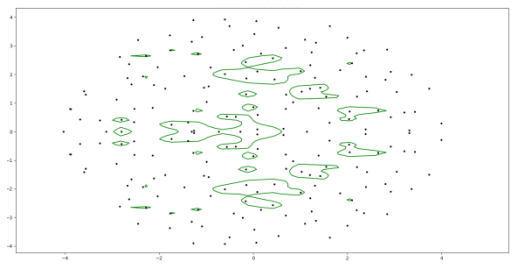
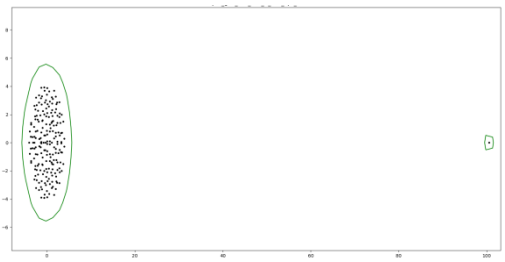
dim = 200, N = 100

λεπτομέρεια (ε = 0.01, 0.1)

ε = 0.01 ε = 1

dim = 200, N = 200

## 4.12. Πίνακες Kahan

Θα κατασκευάσουμε πίνακες Kahan για διαστάσεις από 5 x 5 ως 200 x 200.

Θα υπολογίσουμε και θα απεικονίσουμε σε γράφημα το ψευδοφάσμα του $A$ για διαφορετικές τιμές του ε (0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02) με ακρίβεια 300 καθώς και το χωρίο $\Omega$ που κατασκεύασε ο αλγόριθμος για κάθε περίπτωση.

Συμπεράσματα:

- Σε όλες τις περιπτώσεις το χωρίο $\Omega$ περιλάμβανε ολόκληρο το ψευδοφάσμα (ακόμη και στα γραφήματα που έχουμε κάνει μεγέθυνση και δε φαίνεται).
- Ο υπολογισμός του ψευδοφάσματος με τις παραμέτρους που θέσαμε ήταν ακριβής.
- Οι χρόνοι ήταν παρόμοιοι με αυτούς των τυχαίων πινάκων

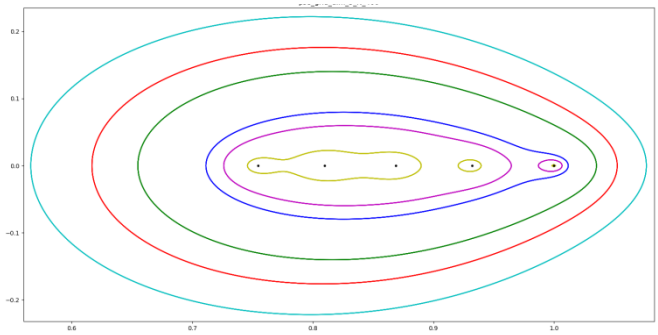
### Κλήση συνάρτησης

```
check_pseudospectra(algorithm = "grid", data = "kahan")
```

## Σύγκριση χωρίων και ψευδοφασμάτων

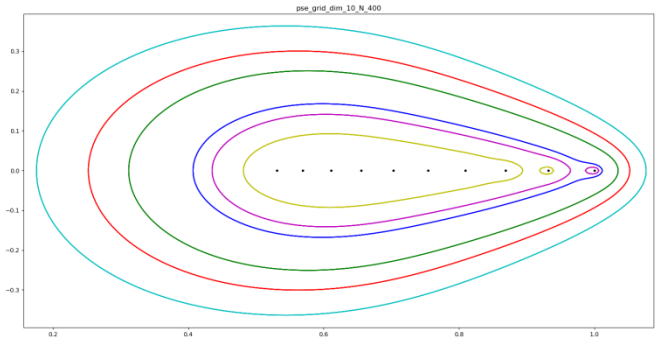
dim = 5, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



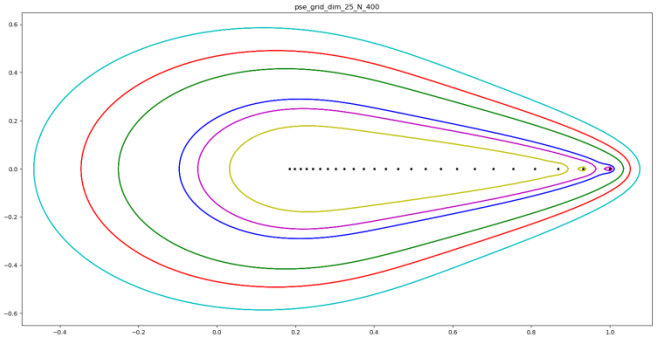
dim = 10, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



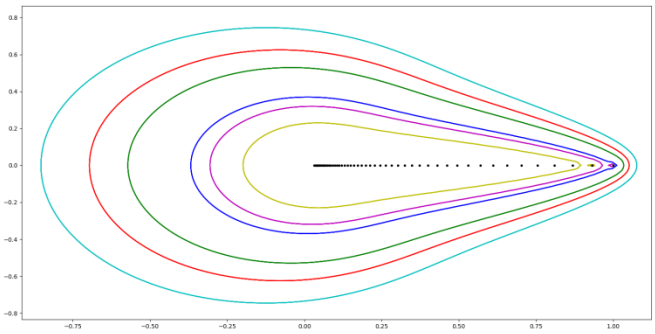

dim = 25, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



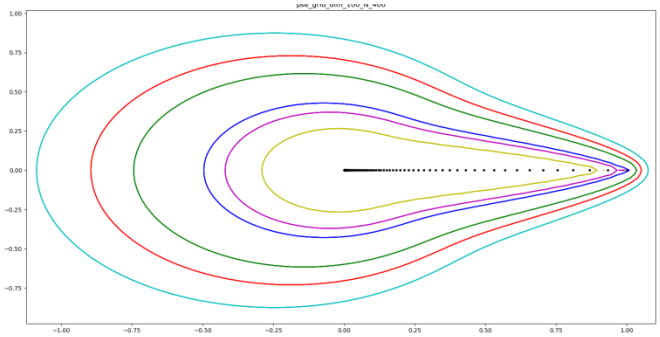
dim = 50, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



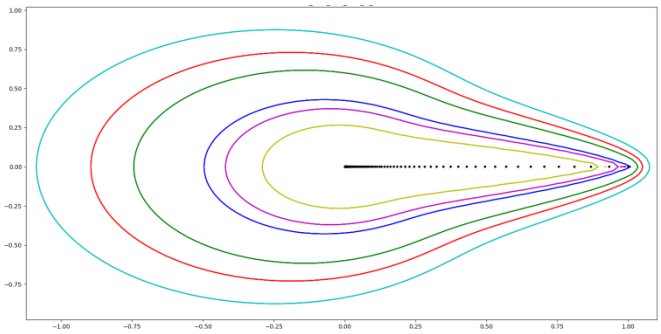
dim = 100, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



dim = 200, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



Χρονομέτρηση

Κλήσησυνάρτησης

```
check_time("grid", "kahan")
```

| dim | N | time | svdtime |
|---|---|---|---|
| 2 | 5 | 0.00 | 0.00 |
| 2 | 10 | 0.01 | 0.01 |
| 2 | 20 | 0.03 | 0.02 |
| 2 | 50 | 0.18 | 0.14 |
| 2 | 100 | 0.72 | 0.59 |
| 2 | 200 | 5.25 | 4.31 |
| 5 | 5 | 0.00 | 0.00 |
| 5 | 10 | 0.01 | 0.01 |
| 5 | 20 | 0.04 | 0.03 |
| 5 | 50 | 0.23 | 0.20 |
| 5 | 100 | 0.91 | 0.79 |
| 5 | 200 | 6.79 | 5.86 |
| 10 | 5 | 0.00 | 0.00 |
| 10 | 10 | 0.02 | 0.02 |
| 10 | 20 | 0.06 | 0.06 |
| 10 | 50 | 0.37 | 0.34 |
| 10 | 100 | 1.47 | 1.34 |
| 10 | 200 | 10.19 | 9.26 |
| 25 | 5 | 0.02 | 0.01 |
| 25 | 10 | 0.06 | 0.06 |
| 25 | 20 | 0.22 | 0.22 |
| 25 | 50 | 1.38 | 1.34 |
| 25 | 100 | 6.12 | 5.96 |
| 25 | 200 | 39.26 | 38.09 |
| 50 | 5 | 0.07 | 0.06 |
| 50 | 10 | 0.23 | 0.23 |
| 50 | 20 | 1.12 | 1.10 |
| 50 | 50 | 6.43 | 6.34 |
| 50 | 100 | 24.23 | 23.95 |
| 50 | 200 | 125.94 | 124.54 |
| 100 | 5 | 0.43 | 0.40 |
| 100 | 10 | 1.64 | 1.62 |
| 100 | 20 | 6.69 | 6.65 |
| 100 | 50 | 47.40 | 47.24 |
| 100 | 100 | 195.86 | 195.31 |
| 100 | 200 | 947.11 | 944.84 |
| 200 | 5 | 1.99 | 1.83 |
| 200 | 10 | 7.34 | 7.20 |
| 200 | 20 | 30.54 | 30.37 |
| 200 | 50 | 219.05 | 218.75 |
| 200 | 100 | 828.37 | 827.63 |
| 200 | 200 | 4048.26 | 4045.51 |

## 4.13. Πίνακες Toeplitz GrCar

Θα κατασκευάσουμε πίνακες Toeplitz GrCar για διαστάσεις από 5 x 5 ως 100 x100.

Θα υπολογίσουμε και θα απεικονίσουμε σε γράφημα το ψευδοφάσμα του $A$ για διαφορετικές τιμές του ε (0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02) με ακρίβεια 300 καθώς και το χωρίο $\Omega$ που κατασκεύασε ο αλγόριθμος για κάθε περίπτωση.

Συμπεράσματα:

- Σε όλες τις περιπτώσεις το χωρίο $\Omega$ περιλάμβανε ολόκληρο το ψευδοφάσμα (ακόμη και στα γραφήματα που έχουμε κάνει μεγέθυνση και δε φαίνεται).
- Ο υπολογισμός του ψευδοφάσματος με τις παραμέτρους που θέσαμε ήταν ακριβής.
- Οι χρόνοι ήταν λίγο μικρότεροι από αυτούς των τυχαίων πινάκων

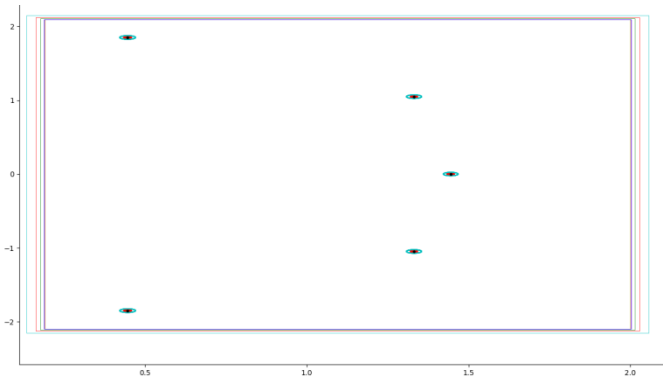
Κλήση συνάρτησης

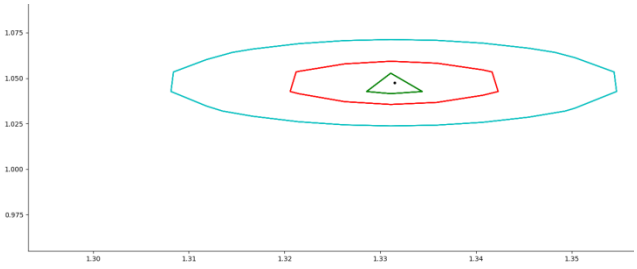
```
check_pseudospectra(algorithm = "grid", data = "grcar")
```

## Σύγκριση χωρίων και ψευδοφασμάτων

dim = 5, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02

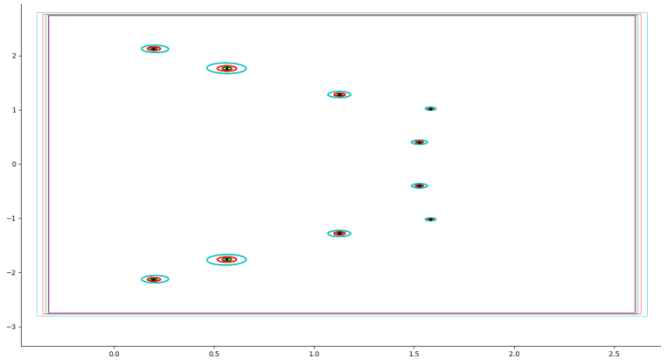


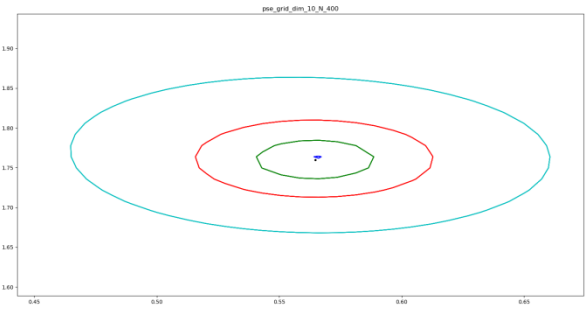


λεπτομέρεια

dim = 10, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02

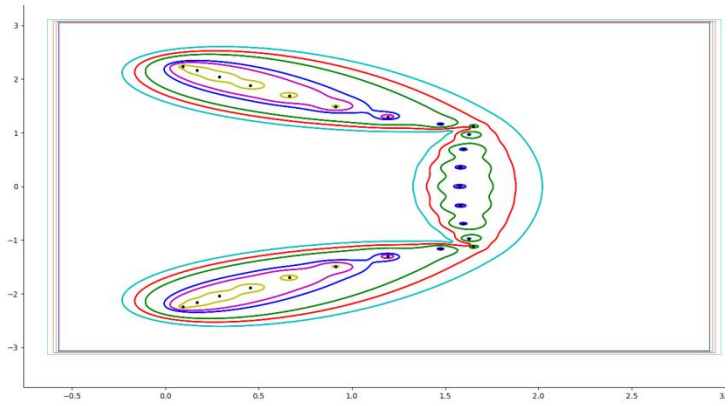




λεπτομέρεια

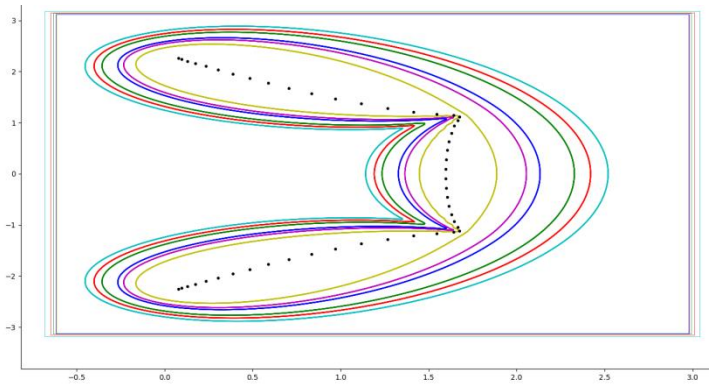
dim = 25, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



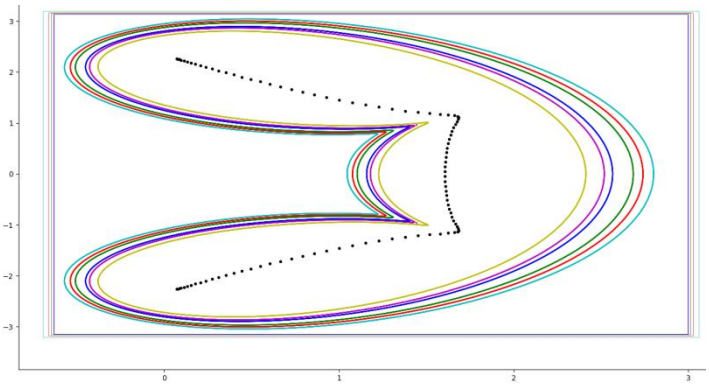
dim = 50, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



dim = 100, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



Χρονομέτρηση

Κλήσησυνάρτησης

```
check_time("grid", "grcar")
```

| dim | N | time | svdtime |
|---|---|---|---|
| 2 | 5 | 0.00 | 0.00 |
| 2 | 10 | 0.01 | 0.01 |
| 2 | 20 | 0.03 | 0.02 |
| 2 | 50 | 0.16 | 0.13 |
| 2 | 100 | 0.65 | 0.51 |
| 2 | 200 | 5.36 | 4.25 |
| 5 | 5 | 0.00 | 0.00 |
| 5 | 10 | 0.01 | 0.01 |
| 5 | 20 | 0.03 | 0.03 |
| 5 | 50 | 0.21 | 0.18 |
| 5 | 100 | 0.83 | 0.70 |
| 5 | 200 | 6.22 | 5.18 |
| 10 | 5 | 0.00 | 0.00 |
| 10 | 10 | 0.01 | 0.01 |
| 10 | 20 | 0.06 | 0.06 |
| 10 | 50 | 0.35 | 0.31 |
| 10 | 100 | 1.64 | 1.48 |
| 10 | 200 | 10.25 | 9.23 |
| 25 | 5 | 0.02 | 0.02 |
| 25 | 10 | 0.07 | 0.07 |
| 25 | 20 | 0.29 | 0.28 |
| 25 | 50 | 1.80 | 1.75 |
| 25 | 100 | 7.87 | 7.64 |
| 25 | 200 | 38.46 | 37.35 |
| 50 | 5 | 0.06 | 0.05 |
| 50 | 10 | 0.22 | 0.21 |
| 50 | 20 | 0.88 | 0.87 |
| 50 | 50 | 5.79 | 5.71 |
| 50 | 100 | 21.52 | 21.34 |
| 50 | 200 | 122.01 | 120.08 |
| 100 | 5 | 0.36 | 0.33 |
| 100 | 10 | 1.44 | 1.41 |
| 100 | 20 | 5.88 | 5.83 |
| 100 | 50 | 35.94 | 35.82 |
| 100 | 100 | 156.65 | 156.09 |
| 100 | 200 | 743.23 | 741.09 |
| 200 | 5 | 2.58 | 2.26 |
| 200 | 10 | 8.48 | 8.23 |
| 200 | 20 | 30.78 | 30.55 |
| 200 | 50 | 193.62 | 193.19 |
| 200 | 100 | 865.37 | 864.52 |
| 200 | 200 | 3643.28 | 3639.74 |

# 5. Ο αλγόριθμος Inclusion-Exclusion: Βελτίωση του Αλγορίθμου GRID

## 5.1. Περιγραφή του Αλγορίθμου Inclusion-Exclusion

Ο αλγόριθμος GRID, για την εύρεση του ψευδοφάσματος ενός πίνακα $A \in \mathbb{C}^{vxv}$, επιλέγει ένα χωρίο του μιγαδικού επιπέδου, και κατασκευάζει ένα πλέγμα $\Omega$ (grid) μέσα στο χωρίο αυτό, που αποτελείται από σημεία $z$ του μιγαδικού επιπέδου. Στη συνέχεια υπολογίζει την$(zI - A)$για κάθε $z \in \Omega$και αποφαίνεται εάν το συγκεκριμένο $z$ ανήκει στο ψευδοφάσμα με βάση τον Ορισμό:

$$\sigma_\varepsilon(A) = \{\lambda \in \mathbb{C} : smin(\lambda I_v - A) < \varepsilon\} ,$$

αναφερόμαστε στη νόρμα $|| o ||_2$, συμβολίζοντας με $smin(\cdot)$ την ελάχιστηιδιάζουσα τιμή ενός πίνακα.

Ο αλγόριθμος GRID για την εύρεση του ψευδοφάσματος για μια τιμή του ε ξεκινά με την επιλογή του χωρίου $\Omega$ και στη συνέχεια, για κάθε σημείο του πλέγματος υπολογίζεται η $s_{min}$ $(zI - A)$,ώστε να προσδιοριστούν τα σημεία του $\Omega$ που ανήκουν στο $\partial\Lambda_\varepsilon$.

Ο αριθμός των SVD παραγοντοποιήσεων αποτελεί το κύριο χρονικό κόστος του. Δεν μπορούν όμως να μειωθούν στη συγκεκριμένη υλοποίηση καθώς ο αλγόριθμος πρέπει να υπολογίζει την $(zI - A)$ για κάθε σημείοτου πλέγματος.

Ο αλγόριθμος GRID είναι εξαιρετικά σταθερός και δίνει σίγουρα σωστά αποτελέσματα, αλλά ταυτόχρονα είναι και εξαιρετικά χρονοβόρος. Το βασικό πρόβλημα είναι η μορφή και το μέγεθος του χρησιμοποιούμενου χωρίου $\Omega$, καθώς αυτό πρέπει να είναι τέτοιο ώστε αφενός να περιέχει ολόκληρο το ψευδοφάσμα για την συγκεκριμένη τιμή του ε και αφετέρου να μην είναι πολύ μεγαλύτερο απ' αυτό, έτσι ώστε να μην γίνονται πολύ περισσότερες SVD παραγοντοποιήσεις από όσες χρειάζονται.

Το ζήτημα είναι, χωρίς να αλλάξει η μέθοδος υπολογισμού τουχωρίου $\Omega$, να μειώσουμε τον αριθμό των σημείων τα οποία θα εξετάσουμε αν ανήκουν ή όχι εντός (ή στο σύνορο) του $\Lambda_\varepsilon$.

Ένας τέτοιος αλγόριθμος είναι ο *Inclusion-Exclusion (IE)* ο οποίος αναπτύχθηκε από τους Ιωάννη Κούτη και Ευστράτιο Γαλλόπουλο. Ξεκινά από ένα χωρίο $\Omega$ το οποίο γνωρίζουμε ότι περιέχει το $\Lambda_\varepsilon$ *(inclusion region)* και στη συνέχεια αποκλείει κάποια σημεία στο πλέγμα του συγκεκριμένου χωρίου. Για τα $z \in \Omega$ που γνωρίζουμε την τριάδα$(s_{min}, u_{min}, v_{min})$ που δίνεται κάνοντας μια SVD παραγοντοποίηση του πίνακα $zI - A$, μπορούμε πιθανώς να κατασκευάσουμε ένα *exclusion region*, δηλαδή μια περιοχή η οποία δεν έχει κανένα κοινό σημείο με το ψευδοφάσμα του $A$.

## 5.2. Ψευδοκώδικας

**ΕΙΣΟΔΟΣ:**

$A \in \mathbb{C}^{vxv}$

$\varepsilon > 0$

N, ο αριθμός των σημείων του πλέγματος σε κάθε άξονα

**ΒΗΜΑ 1:** Ορισμός των ορίων του πλέγματος και των βημάτων

$x_{min} = \lambda_{min}(\mathbf{A}_H) - \varepsilon \|\mathbf{A}\|_2$

$x_{max} = \lambda_{max}(\mathbf{A}_H) - \varepsilon \|\mathbf{A}\|_2$

$y_{min} = \lambda_{min}(\mathbf{A}_{SH}) - \varepsilon \|\mathbf{A}\|_2$

$y_{max} = \lambda_{max}(\mathbf{A}_{SH}) - \varepsilon \|\mathbf{A}\|_2$

$x_{step} = (x_{max} - x_{min}) / N$

$y_{step} = (y_{max} - y_{min}) / N$

**ΒΗΜΑ 2:**

Για κάθε σημείο (x, y) του πλέγματος:

$z = x + iy$

$s = s_{min}(zI - \mathbf{A})$

Τέλος Επανάληψης

Για κάθε σημείο (x, y) του πλέγματος:

Αν $s > \varepsilon$

Για κάθε $z' \in D°(z, s-\varepsilon)$:

z': out

$\Omega = \Omega \setminus D°(z, s - \varepsilon)$

Τέλος_επανάληψη ς

Τέλος_αν

Τέλος_επανάληψης

Σχεδιάζονται τα σημεία εκείνα για τα οποία ισχύει $s_{min}(zI - \mathbf{A}) = \varepsilon$

**ΤΕΛΟΣ**

Για τον υπολογισμό του $\Lambda_\varepsilon$ για περισσότερα ε, ο αλγόριθμος εκτελεί την παραπάνω διαδικασία μόνο για το μεγαλύτερο εξ' αυτών και απλώς εκτελείται η τελευταία εντολή για κάθε τιμή [ΔΕΣΥΠΡΗ2018].

## 5.3. Κώδικας Matlab

Στηρίζεται σε κώδικα της διπλωματικής εργασίας [ΔΕΣΥΠΡΗ2018].

### file:inex_01_program.m

```
A = [   0.8147 0.0975 0.1576 0.1419 0.6557 ;
        0.9058 0.2785 0.9706 0.4218 0.0357 ;
        0.1270 0.5469 0.9572 0.9157 0.8491 ;
        0.9134 0.9575 0.4854 0.7922 0.9340 ;
        0.6324 0.9649 0.8003 0.9595 0.6787
    ];

N = 10;

epsilons = [1  0.8  0.5  0.2  0.005];
[X, Y, Z, Vxmin, Vxmax, Vymin, Vymax, maxepsilon] = inexcalculate(A, epsilons, N);
gridplot(X, Y, Z, epsilons, Vxmin, Vxmax, Vymin, Vymax);
```

### file:  gridplot.m

```
function gridplot(X, Y, Z, epsilons, Vxmin, Vxmax, Vymin, Vymax)
figure(1);   %% create a figure
    hold on;
    mycolour = "krgbm";

    [maxepsilon, ind] = max(epsilons);
    [axisxmin, axisxmax, axisymin, axisymax] =
grid_find_axis_limits(Vxmin(ind), Vxmax(ind), Vymin(ind), Vymax(ind));

axis([axisxmin axisxmax axisymin axisymax]);
          %% Change the axis limits so that the x-axis ranges from axisxmin to axisxmax and the y-axis ranges from axisymin to axisymax

    for i = 1:length(epsilons)
        %% Plotting the grid, black color
        tax = [Vxmin(i) Vxmax(i) Vxmax(i) Vxmin(i) Vxmin(i)];  %% x points for a rectangle
        tay = [Vymin(i) Vymin(i) Vymax(i) Vymax(i) Vymin(i)];  %% y points for a rectangle
        p = plot(tax, tay, mycolour(mod(i, length(mycolour)) + 1) );
%% plots Omega rectangle;    plot() plots a 2D line,
                                %% getting points from the corresponding values of the tax and tay vectors

        hold on;  %% hold on retains plots in the current axes so that new plots added do not delete existing plots

contour(X, Y, Z', [epsilons(i) epsilons(i)], mycolour(mod(i, length(mycolour)) + 1), "linewidth", 2) ;
             %% Display the contours of the Z', at Z' = epsilon
    end
    %%  pause;
    %%  print("grid_02_merged_matlab.jpg", "-djpg") ;   %% print the figure in the file
```

file:  inexcalculate.m

```
function [X, Y, Z, Vxmin, Vxmax, Vymin, Vymax, maxepsilon] = inexcalculate(A, epsilons, N)

print_2d_array(A, "A");

n = length(A);
norma = norm(A,2);  %% norm(X, 2) returns the 2-norm or maximum singular value of matrix X,
                    %% which is approximately max(svd(X)).
                    %% This value is equivalent to norm(X) in Matlab, but not in Python numpy.

fprintf("matrix A:   rows = %d,   2-norm = %f\n\n", n, norma);

Ah =  (A+A')/2;     %% A' is the transpose matrix
Ash = (A-A')/2;

print_2d_array(Ah, "Ah");

print_2d_array(Ash, "Ash");

%% The eigenvalues of Ah and Ash in ascending order
EAh =  sort(real(eig(Ah)));     %% eig(A) returns a column vector containing
EAsh = sort(imag(eig(Ash)));    %% the eigenvalues of square matrix A.

print_1d_array(EAh, "EAh");

print_1d_array(EAsh, "EAsh");

for metr = 1:length(epsilons)
    Vxmin(metr) = EAh(1) - epsilons(metr) * norma;
    Vxmax(metr) = EAh(end) + epsilons(metr) * norma;
    Vymin(metr) = EAsh(1) - epsilons(metr) * norma;
    Vymax(metr) = EAsh(end) + epsilons(metr) * norma;
    Vxstep(metr) = (Vxmax(metr) - Vxmin(metr)) / N;
    Vystep(metr) = (Vymax(metr) - Vymin(metr)) / N;
    fprintf("epsilons(%d) = %f,    xmin = %f,    xmax = %f,    ymin= %f,    ymax = %f,    xstep = %f,     ystep = %f \n\n\n",
             metr, epsilons(metr), Vxmin(metr), Vxmax(metr), Vymin(metr), Vymax(metr), Vxstep(metr), Vystep(metr));
end

[maxepsilon, indexofmaxepsilon] = max(epsilons);
xmin = Vxmin(indexofmaxepsilon);
xmax = Vxmax(indexofmaxepsilon);
ymin = Vymin(indexofmaxepsilon);
ymax = Vymax(indexofmaxepsilon);
xstep = Vxstep(indexofmaxepsilon);
ystep = Vystep(indexofmaxepsilon);

fprintf("xmin = %f,    xmax = %f,    ymin= %f,    ymax = %f,    xstep = %f,     ystep = %f \n\n\n",
          xmin, xmax, ymin, ymax, xstep, ystep);
```

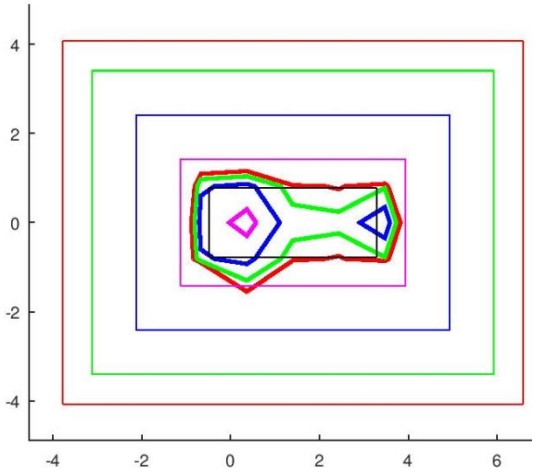
```
Z = zeros(N,N);

for I=1:N
    x = xmin + (I-1)*xstep;
    X(I)=x;
    for J=1:N
        y = ymin + (J-1)*ystep;
Y(J)=y;
z = x + i*y;
zita(I,J) = z;
end
end

svdcount = 0;

for I=1:N
    for J=1:N
        if (Z(I,J) == 0)
            svdcount = svdcount + 1 ;
            S = svd(zita(I,J)*eye(n)-A);
            smin = S(end);
            Z(I,J) = smin;

            %Perform the exclusion
            if (smin > maxepsilon)
                x = real(zita(I,J));
y = imag(zita(I,J));
                %%  plot(x, y, 'b+')
%%  hold on
                %%  circle(x,y,smin-maxepsilon);
                for a=1:N
                    for b=1:N
                        if (abs(zita(a,b)-zita(I,J))<(smin-maxepsilon))
                            Z(a,b)=Z(I,J);
                        end
                    end
                end
            end
        end
    end
end

fprintf("Number of svd factorializations = %d out of %d \n\n", svdcount, N*N);

fprintf("Results for max epsilon = %f \n\n", maxepsilon);

print_1d_array(X, "X");

print_1d_array(Y, "Y");

print_2d_array(Z, "Z");

print_2d_array(Z', "Z'");

end
```

## 5.4. Αποτελέσματα του Κώδικα Matlab

```
matrix A:   rows = 5,   cols = 5,   total elements = 25

      0.8147      0.0975      0.1576      0.1419      0.6557
      0.9058      0.2785      0.9706      0.4218      0.0357
      0.1270      0.5469      0.9572      0.9157      0.8491
      0.9134      0.9575      0.4854      0.7922      0.9340
      0.6324      0.9649      0.8003      0.9595      0.6787


matrix A:   rows = 5,   2-norm = 3.312915


matrix Ah:   rows = 5,   cols = 5,   total elements = 25

      0.8147      0.5017      0.1423      0.5276      0.6441
      0.5017      0.2785      0.7588      0.6896      0.5003
      0.1423      0.7588      0.9572      0.7006      0.8247
      0.5276      0.6896      0.7006      0.7922      0.9467
      0.6441      0.5003      0.8247      0.9467      0.6787


matrix Ash:   rows = 5,   cols = 5,   total elements = 25

      0.0000     -0.4042      0.0153     -0.3857      0.0116
      0.4042      0.0000      0.2118     -0.2679     -0.4646
     -0.0153     -0.2118      0.0000      0.2151      0.0244
      0.3857      0.2679     -0.2151      0.0000     -0.0127
     -0.0116      0.4646     -0.0244      0.0127      0.0000


vector EAh:    number of elements = 5

     -0.4627     -0.0870      0.0740      0.7339      3.2631


vector EAsh:    number of elements = 5

     -0.7575     -0.3439      0.0000      0.3439      0.7575

epsilons(1) = 1.000000,    xmin = -3.775599,    xmax = 6.576030,    ymin= -4.070389,    ymax = 4.070389,    xstep = 1.035163,
ystep = 0.814078

epsilons(2) = 0.800000,    xmin = -3.113016,    xmax = 5.913447,    ymin= -3.407806,    ymax = 3.407806,    xstep = 0.902646,
ystep = 0.681561

epsilons(3) = 0.500000,    xmin = -2.119141,    xmax = 4.919572,    ymin= -2.413932,    ymax = 2.413932,    xstep = 0.703871,
ystep = 0.482786

epsilons(4) = 0.200000,    xmin = -1.125267,    xmax = 3.925698,    ymin= -1.420057,    ymax = 1.420057,    xstep = 0.505097,
ystep = 0.284011

epsilons(5) = 0.005000,    xmin = -0.479249,    xmax = 3.279680,    ymin= -0.774039,    ymax = 0.774039,    xstep = 0.375893,
ystep = 0.154808

xmin = -3.775599,    xmax = 6.576030,    ymin= -4.070389,    ymax = 4.070389,    xstep = 1.035163,    ystep = 0.814078
```

**Number of svd factorializations = 38 out of 100**

```
Results for max epsilon = 1.000000

vector X:    number of elements = 10

     -3.7756     -2.7404     -1.7053     -0.6701      0.3651      1.4002      2.4354      3.4705      4.5057      5.5409

vector Y:    number of elements = 10

     -4.0704     -3.2563     -2.4422     -1.6282     -0.8141      0.0000      0.8141      1.6282      2.4422      3.2563


matrix Z:   rows = 10,   cols = 10,   total elements = 100

      4.9637      4.9637      4.9637      3.3491      3.3491      3.9755      3.9755      3.9755      3.9755      3.9755
      4.9637      4.9637      4.9637      3.3491      3.3491      3.9755      3.9755      3.9755      3.9755      3.9755
      3.4150      3.4150      4.9637      4.9637      3.3491      3.3491      3.9755      3.9755      3.9755      3.9755
      3.4150      3.4150      3.4150      4.9637      0.5376      0.3932      0.5376      1.8841      1.8841      1.8841
      3.4150      3.4150      3.4150      1.0650      0.4085      0.0802      0.4085      1.8292      2.1412      3.0739
      3.9350      3.9350      3.9350      2.3477      0.9378      0.6669      0.9378      3.0739      3.0739      3.0739
      3.9350      3.9350      3.9350      3.9350      1.0257      0.7042      1.0257      3.0739      3.0739      3.0739
      3.9350      3.9350      3.9350      3.9350      0.8274      0.2577      0.8274      3.3274      3.3274      3.3274
      3.9350      3.9350      2.8038      2.8038      2.8038      2.4362      3.3274      3.3274      3.3274      3.3274
      3.9350      2.8038      2.8038      2.8038      2.8038      2.4362      3.3274      3.3274      3.3274      3.3274


matrix Z':   rows = 10,   cols = 10,   total elements = 100

4.9637      4.9637      3.4150      3.4150      3.4150      3.9350      3.9350      3.9350      3.9350      3.9350
      4.9637      4.9637      3.4150      3.4150      3.4150      3.9350      3.9350      3.9350      3.9350      2.8038
      4.9637      4.9637      4.9637      3.4150      3.4150      3.9350      3.9350      3.9350      2.8038      2.8038
      3.3491      3.3491      4.9637      4.9637      1.0650      2.3477      3.9350      3.9350      2.8038      2.8038
      3.3491      3.3491      3.3491      0.5376      0.4085      0.9378      1.0257      0.8274      2.8038      2.8038
      3.9755      3.9755      3.3491      0.3932      0.0802      0.6669      0.7042      0.2577      2.4362      2.4362
      3.9755      3.9755      3.9755      0.5376      0.4085      0.9378      1.0257      0.8274      3.3274      3.3274
      3.9755      3.9755      3.9755      1.8841      1.8292      3.0739      3.0739      3.3274      3.3274      3.3274
      3.9755      3.9755      3.9755      1.8841      2.1412      3.0739      3.0739      3.3274      3.3274      3.3274
      3.9755      3.9755      3.9755      1.8841      3.0739      3.0739      3.0739      3.3274      3.3274      3.3274
```



## 5.5. Βελτίωση του τρόπου που γίνεται το exclusion

Στην αρχική εκδοχή του αλγόριθμου Inclusion-Exclusion, το exclusion γινόταν με ένα διπλό loop που έτρεχε κάθε φορά που γινόταν κάποιο exclusion, ώστε να βρεθούν τα στοιχεία του πίνακα Zπου πρέπει να εξαχθούν από την εξέταση στη συνέχεια. Αυτό οδηγεί σε πολυπλοκότητα χρόνου exclusion $O(N^2)$ όπου N είναι η ακρίβεια (πλήθος σημείων ανά άξονα).

Βελτιώσαμε το exclusion βάζοντας τα στοιχεία του Z μαζί με τα indices σε μια μονοδιάστατη λίστα. Κάθε φορά που γίνεται κάποιο exclusion, βγάζουμε τα στοιχεία του πίνακα Z που πρέπει να εξαχθούν από την εξέταση από τη λίστα, ώστε να μην τα εξετάσουμε στη συνέχεια. Αυτό οδηγεί σε πολυπλοκότητα χρόνου exclusion O(N). Στις χρονομετρήσεις που ακολουθούν βλέπουμε ότι το exclusion time με βάση τον αρχικό τρόπο εκτοξεύεται καθώς πάμε από ακρίβεια 100 σε 200, ενώ με τον βελτιωμένο τρόπο αυξάνεται γραμμικά (200 x 200 ακρίβεια έχει 4πλάσια σημεία από την 100 x 100).

Χρονομετρήσεις

Αρχικός τρόπος:

algorithm: inexold, dim = 100, N = 100, time = 15.701841 seconds, svdtime = 12.930729 seconds, svdcount = 969 out of 10000, exclusiontime = 2.638437 seconds

algorithm: inexold, dim = 200, N = 200, time = 225.695099 seconds, svdtime = 142.285086 seconds, svdcount = 2066 out of 40000, exclusiontime = 82.901686 seconds

Βελτιωμένος τρόπος:

algorithm: inex, dim = 100, N = 100, time = 21.210351 seconds, svdtime = 13.125244 seconds, svdcount = 969 out of 10000, exclusiontime = 7.943380 seconds

algorithm: inex, dim = 200, N = 200, time = 156.936674 seconds, svdtime = 131.624937 seconds, svdcount = 2066 out of 40000, exclusiontime = 24.759557 seconds

## 5.6. Συνάρτηση inexcalculate

```
def inexcalculate(A, epsilons, N):
    svdtime = 0
    exclusiontime = 0
    n, cols = A.shape
    norma = LA.norm(A, 2)     ##norm(X, 2) returns the 2-norm or maximum singular value of matrix X,
                              ## which is approximately max(svd(X)).
                              ## This value is equivalent to norm(X) in Matlab, but not in Python numpy.

    Ah = (A + A.transpose()) / 2
    Ash = (A - A.transpose()) / 2

    EAh, koko =  LA.eig(Ah)    ## LA.eig(Ah) returns a column vector containing
    EAh = sorted(EAh.real)     ## the eigenvalues of square matrix A.
    EAh = np.array(EAh)
    EAsh, koko =  LA.eig(Ash)
    EAsh = sorted(EAsh.imag)
    EAsh = np.array(EAsh)

    Vxmin = np.zeros((epsilons.size, 1))
    Vxmax = np.zeros((epsilons.size, 1))
    Vymin = np.zeros((epsilons.size, 1))
    Vymax = np.zeros((epsilons.size, 1))
    Vxstep = np.zeros((epsilons.size, 1))
    Vystep = np.zeros((epsilons.size, 1))
    for metr in range(epsilons.size):
        Vxmin[metr] = EAh[0] - epsilons[metr]*norma
        Vxmax[metr] = EAh[-1] + epsilons[metr]*norma
        Vymin[metr] = EAsh[0] - epsilons[metr]*norma
        Vymax[metr] = EAsh[-1] + epsilons[metr]*norma
        Vxstep[metr] = (Vxmax[metr]-Vxmin[metr])/N
        Vystep[metr] = (Vymax[metr]-Vymin[metr])/N

    maxepsilon, indexofmaxepsilon = np.amax(epsilons), np.argmax(epsilons);
    xmin = Vxmin[indexofmaxepsilon]
    xmax = Vxmax[indexofmaxepsilon]
    ymin = Vymin[indexofmaxepsilon]
    ymax = Vymax[indexofmaxepsilon]
    xstep = Vxstep[indexofmaxepsilon]
    ystep = Vystep[indexofmaxepsilon]

    X = np.zeros((N, 1))
    Y = np.zeros((N, 1))
    Z = np.zeros((N, N))
    zita = np.zeros((N, N), dtype=np.complex_)
```

```
    for I in range(1, N+1):
        x = xmin + (I-1) * xstep
        X[I-1] = x
        for J in range(1, N+1):
y = ymin + (J-1) * ystep
            Y[J-1] = y
z = complex(x, y)
zita[I-1][J-1] = z

svdcount = 0

    zita1d = []
    for i in range(N):
        for j in range(N):
            zita1d.append((i, j, zita[i][j]))

    for I in range(1, N+1):
        for J in range(1, N+1):
            if (Z[I-1][J-1] == 0):
                svdcount = svdcount + 1
                currenttime = time.time()
                u, S, vh = np.linalg.svd(zita[I-1][J-1] * np.eye(n, dtype=int) - A, full_matrices=True)
                svdtime += time.time() - currenttime
                Z[I-1][J-1] = S[-1]
                smin = S[-1]
                if (smin > maxepsilon):
                    currenttime = time.time()
                    listofindicestodelete = []
                    for c in range(len(zita1d)):
                            if ( abs( zita1d[c][2] - zita[I-1][J-1] ) < (smin - maxepsilon) ):
                                Z[zita1d[c][0]][zita1d[c][1]] = Z[I-1][J-1]
listofindicestodelete.append(c)
                    for index in sorted(listofindicestodelete, reverse=True):
                        ## reverse order doesn't throw off the subsequent indexes!!
                        del zita1d[index]
                    exclusiontime += time.time() - currenttime

    return (X, Y, Z, Vxmin, Vxmax, Vymin, Vymax, maxepsilon, svdtime, svdcount, exclusiontime)
```

## 5.7. Python check_pseudospectra σε Πίνακες Τυχαίων Αριθμών

Θα κατασκευάσουμε πίνακες τυχαίων αριθμών για διαστάσεις **dim** από 5x5 ως 100x100.

Θα κρατήσουμε σταθερή την ακρίβεια **N** (πλήθος σημείων του πλέγματος ανά άξονα) στο 50.

Θα υπολογίσουμε και θα απεικονίσουμε σε γράφημα το ψευδοφάσμα του Άγια διαφορετικές τιμές του ε (0.05, 0.2, 0.5, 0.8, 1), **epsilons**, καθώς και το χωρίο Ω που κατασκεύασε ο αλγόριθμος για κάθε περίπτωση.

Συμπεράσματα:

- Σε όλες τις περιπτώσεις το χωρίο Ω περιλάμβανε ολόκληρο το ψευδοφάσμα (ακόμη και στα γραφήματα που έχουμε κάνει μεγέθυνση και δε φαίνεται).
- Ο υπολογισμός του ψευδοφάσματος χρειάζεται μεγαλύτερη ακρίβεια όσο μεγαλώνει η διάσταση, και όσο μικραίνει το ε.

Κλήση συνάρτησης

```
check_pseudospectra(algorithm = "inex", data = "random")
```

Output

```
algorithm: inex, dim = 5, N = 50, time = 0.281189 seconds, svdtime = 0.060560 seconds, svdcount = 772 out of 2500, exclusiontime = 0.194108 seconds
algorithm: inex, dim = 10, N = 50, time = 0.181621 seconds, svdtime = 0.035007 seconds, svdcount = 238 out of 2500, exclusiontime = 0.115588 seconds
algorithm: inex, dim = 25, N = 50, time = 0.170111 seconds, svdtime = 0.061054 seconds, svdcount = 112 out of 2500, exclusiontime = 0.078549 seconds
algorithm: inex, dim = 50, N = 50, time = 0.205637 seconds, svdtime = 0.114075 seconds, svdcount = 63 out of 2500, exclusiontime = 0.058550 seconds
algorithm: inex, dim = 100, N = 50, time = 0.677109 seconds, svdtime = 0.563546 seconds, svdcount = 42 out of 2500, exclusiontime = 0.054026 seconds
algorithm: inex, dim = 200, N = 50, time = 1.774977 seconds, svdtime = 1.540344 seconds, svdcount = 29 out of 2500, exclusiontime = 0.033498 seconds
```

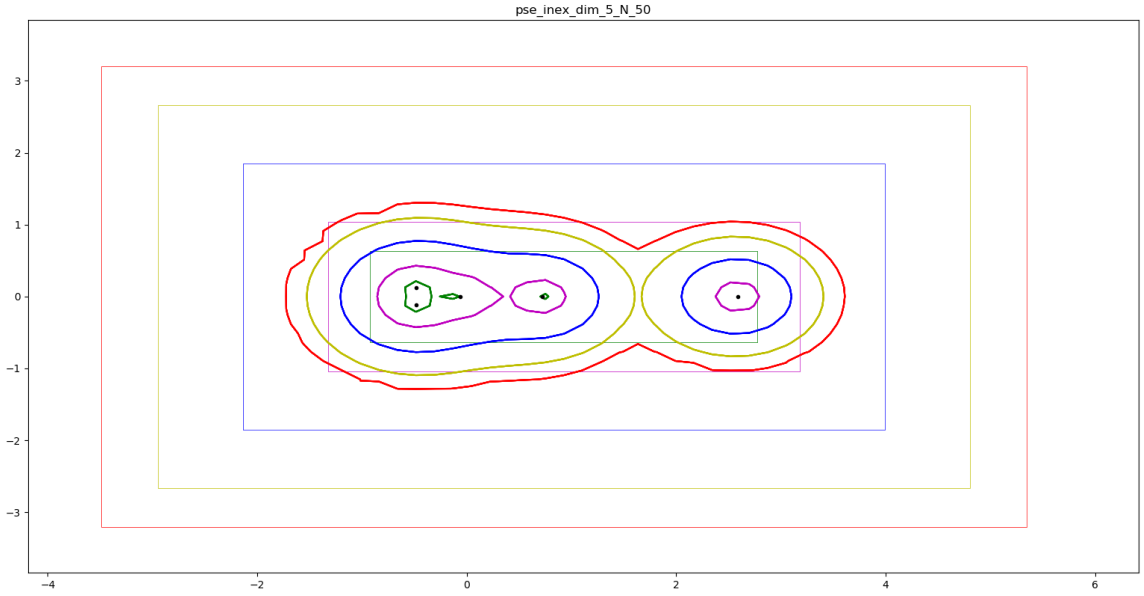
Μετά από κάθε γραμμή εμφανίζεται στην οθόνη το αντίστοιχο γράφημα

N = 50 για όλα τα γραφήματα

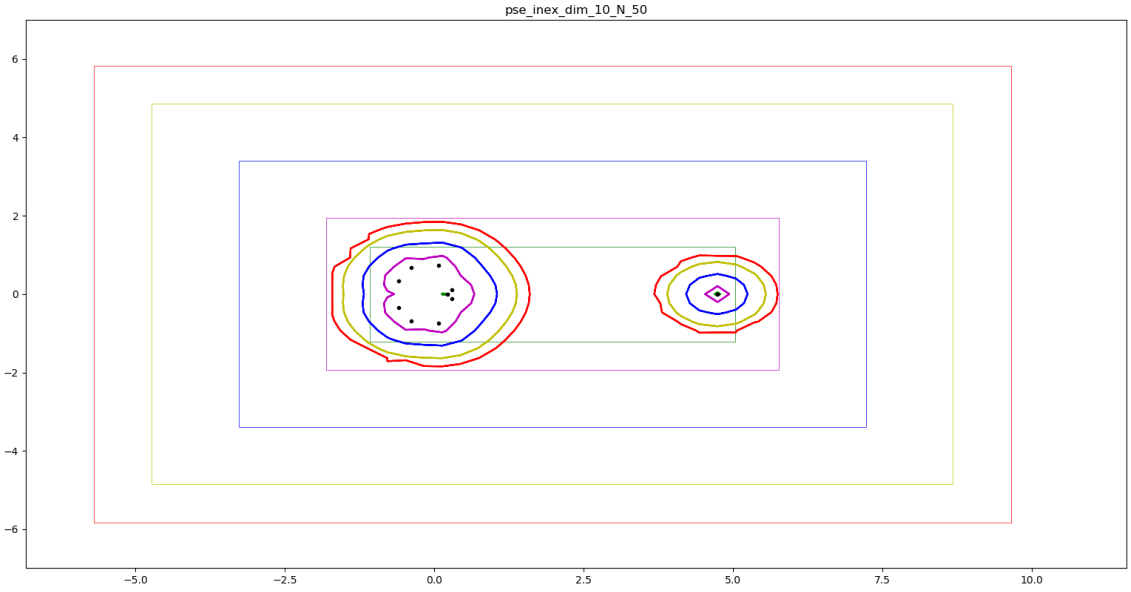
κόκκινο: ε = 1, πράσινο: ε = 0.8, μπλε: ε = 0.5,

ματζέντα: ε = 0.2, κίτρινο: ε = 0.05

dim = 5




dim = 10

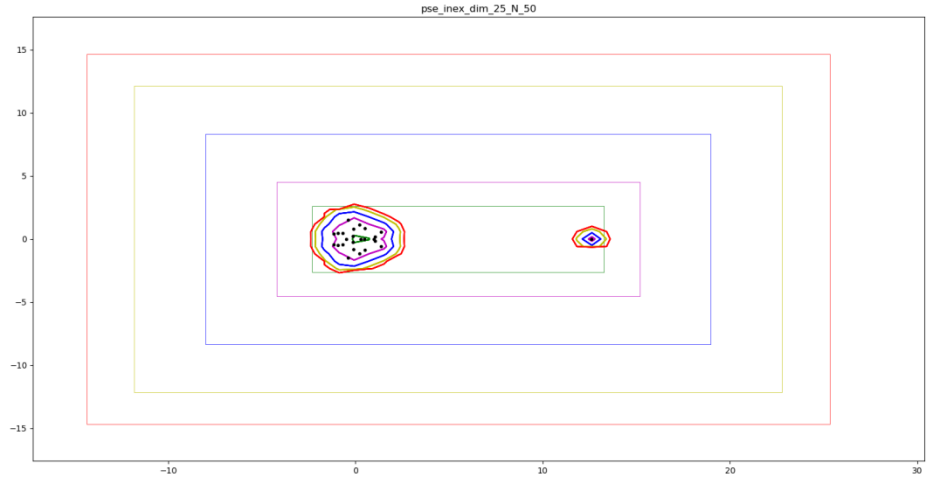


N = 50 για όλα τα γραφήματα

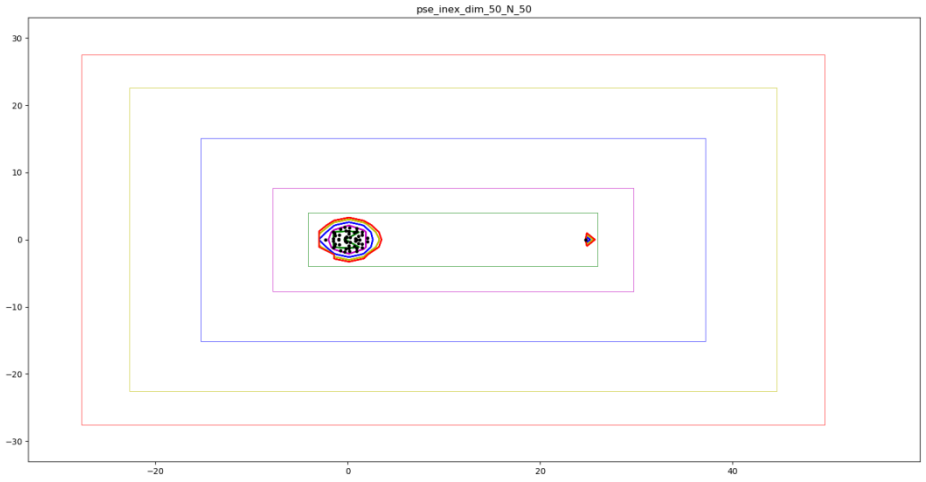
κόκκινο: ε = 1, πράσινο: ε = 0.8, μπλε: ε = 0.5,

ματζέντα: ε = 0.2, κίτρινο: ε = 0.05

dim = 25

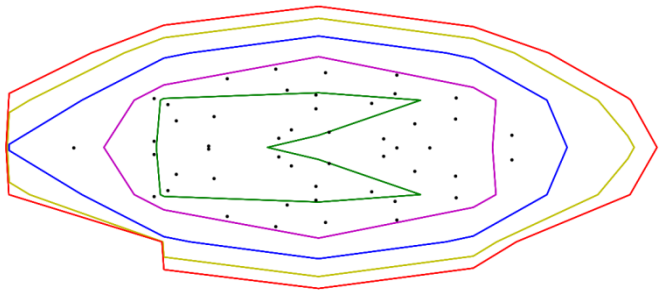



dim = 50




λεπτομέρεια

η μακρινή
δεξιά ιδιοτιμή
δεν μπήκε
στο ψευδοφάσμα



N = 50 για όλα τα γραφήματα

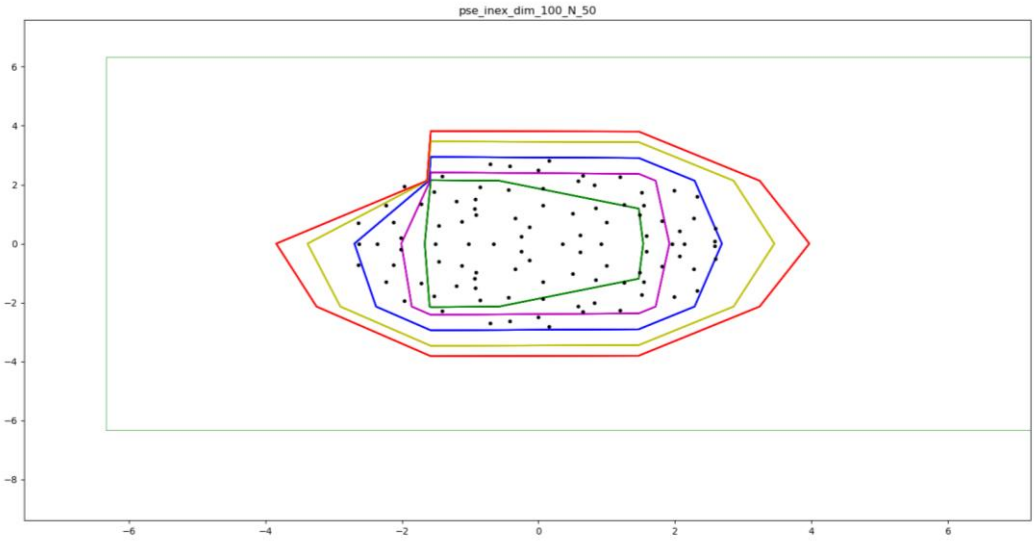
κόκκινο: ε = 1, πράσινο: ε = 0.8, μπλε: ε = 0.5,

ματζέντα: ε = 0.2, κίτρινο: ε = 0.05

dim = 100

λεπτομέρεια

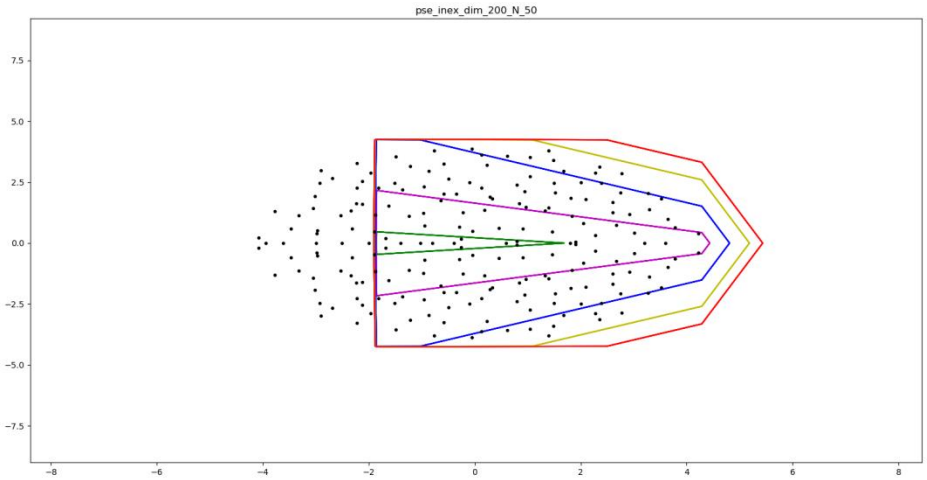
η μακρινή δεξιά ιδιοτιμή δεν μπήκε στο ψευδοφάσμα




dim = 200

λεπτομέρεια

η μακρινή δεξιά ιδιοτιμή δεν μπήκε στο ψευδοφάσμα



## 5.8. Συνάρτηση check_time σε πίνακες τυχαίων αριθμών για χρονομέτρηση

Θα κατασκευάσουμε πίνακες τυχαίων αριθμών για διαστάσεις από 2 x 2 ως200 x 200.

Θα κρατήσουμε σταθερό το ε = 1.

Θα υπολογίσουμε το ψευδοφάσμα του A για διαφορετικές τιμές της ακρίβειας,
ΑπόN = 5 ωςN = 200.

Συμπεράσματα:

Όσον αφορά τον χρόνο εκτέλεσης του αλγόριθμου grid, συμπεραίνουμε ότι ο **χρόνος εκτέλεσης είναι γραμμικός σε σχέση με το μέγεθος του προβλήματος**.

**Το μέγεθος του προβλήματος εξαρτάται από δύο παράγοντες:**

- **τη διάσταση dim του δισδιάστατου πίνακα** (dimxdim) που μελετάμε
- **την ακρίβεια,** δηλαδή τον αριθμό N των σημείων που μελετάμε κατά x και κατά y (NxN συνολικά)

**Το μέγεθος του προβλήματος δεν εξαρτάται από την επιλογή του epsilon** όπως είδαμε σε προηγούμενη ενότητα.

Τονίζουμε ότι **πρόκειται για τετραγωνικούς πίνακες,** οπότε όταν περνάμε, για παράδειγμα, από διάσταση 50 σε διάσταση 100, το μέγεθος του προβλήματος τετραπλασιάζεται, καθώς ο πίνακας από 50 x 50 γίνεται 100 x 100. Το ίδιο ισχύει και με την ακρίβεια, καθώς το grid σημείων που παίρνουμε είναι δισδιάστατο, κατά x και κατά y.

**Όταν το μέγεθος αυξάνεται, σχεδόν όλος ο χρόνος αφορά τις svd παραγοντοποιήσεις. Αυτό συμβαίνει γιατί με τη** χρήση των κατάλληλων δομών δεδομένων, ο χρόνος για το exclusion γίνεται πιθανότατα λογαριθμικός (σίγουρα λιγότερο από γραμμικός).

Κώδικας

Η συνάρτηση **check_time** του κεφαλαίου για τον αλγόριθμο grid.

Κλήση συνάρτησης check_time

```
check_time("inex", "random")
```

Τα αποτελέσματα της χρονομέτρησης του αλγορίθμου inex σε πίνακες τυχαίων αριθμών εμφανίζονται στην επόμενη σελίδα. Επιλέγουμε μερικά για την ανάλυση που ακολουθεί.

| dim | N | grid time | inex time | svd time | svds made | total svds N x N | exclusion time |
|---|---|---|---|---|---|---|---|
| 25 | 50 | 2.20 | 0.16 | 0.06 | 110 | 2500 | 0.08 |
| 25 | 100 | 8.37 | 0.91 | 0.18 | 307 | 10000 | 0.64 |
| 25 | 200 | 46.51 | 6.38 | 0.56 | 1025 | 40000 | 5.49 |
| 50 | 50 | 6.57 | 0.19 | 0.11 | 61 | 2500 | 0.06 |
| 50 | 100 | 25.42 | 0.91 | 0.30 | 161 | 10000 | 0.52 |
| 50 | 200 | 129.56 | 5.25 | 0.87 | 476 | 40000 | 4.03 |
| 100 | 50 | 41.51 | 0.67 | 0.57 | 43 | 2500 | 0.06 |
| 100 | 100 | 201.97 | 1.76 | 1.28 | 94 | 10000 | 0.37 |
| 100 | 200 | 981.37 | 6.52 | 3.43 | 247 | 40000 | 2.73 |
| 200 | 50 | 208.51 | 1.88 | 1.67 | 30 | 2500 | 0.04 |
| 200 | 100 | 856.82 | 3.67 | 3.17 | 58 | 10000 | 0.26 |
| 200 | 200 | 4143.22 | 10.08 | 7.52 | 139 | 40000 | 2.07 |

Με τη χρήση των κατάλληλων δομών δεδομένων, ο χρόνος για το exclusion είναι πλέον μικρός, και γίνεται λιγότερο σημαντικός όσο η διάσταση μεγαλώνει.

Ο χρόνος για τις svd παραγοντοποιήσεις μειώνεται πολύ όσο η διάσταση μεγαλώνει, και όσο η ακρίβεια (πλήθος σημείων ανά άξονα (N) μεγαλώνει. Το ίδιο συμβαίνει και με τον συνολικό χρόνο.

**Χρονομέτρηση του αλγόριθμου inex σε πίνακες τυχαίων αριθμών**

| dim | N | grid time | inex time | svd time | svds made | total svds N x N | exclusion time |
|---|---|---|---|---|---|---|---|
| 2 | 5 | 0.00 | 0.00 | 0.00 | 23 | 25 | 0.00 |
| 2 | 10 | 0.01 | 0.01 | 0.01 | 78 | 100 | 0.00 |
| 2 | 20 | 0.04 | 0.05 | 0.02 | 257 | 400 | 0.02 |
| 2 | 50 | 0.24 | 0.47 | 0.09 | 1397 | 2500 | 0.35 |
| 2 | 100 | 0.84 | 3.24 | 0.30 | 5272 | 10000 | 2.83 |
| 2 | 200 | 4.87 | 25.45 | 1.19 | 20447 | 40000 | 23.81 |
| 5 | 5 | 0.00 | 0.00 | 0.00 | 16 | 25 | 0.00 |
| 5 | 10 | 0.01 | 0.01 | 0.00 | 42 | 100 | 0.00 |
| 5 | 20 | 0.05 | 0.03 | 0.01 | 126 | 400 | 0.01 |
| 5 | 50 | 0.27 | 0.27 | 0.05 | 618 | 2500 | 0.20 |
| 5 | 100 | 1.08 | 1.89 | 0.17 | 2242 | 10000 | 1.62 |
| 5 | 200 | 5.56 | 14.09 | 0.69 | 8499 | 40000 | 13.03 |
| 10 | 5 | 0.01 | 0.00 | 0.00 | 13 | 25 | 0.00 |
| 10 | 10 | 0.02 | 0.01 | 0.00 | 29 | 100 | 0.00 |
| 10 | 20 | 0.08 | 0.03 | 0.01 | 76 | 400 | 0.01 |
| 10 | 50 | 0.47 | 0.20 | 0.04 | 291 | 2500 | 0.14 |
| 10 | 100 | 1.95 | 1.31 | 0.13 | 959 | 10000 | 1.09 |
| 10 | 200 | 10.08 | 10.00 | 0.47 | 3440 | 40000 | 9.19 |
| 25 | 5 | 0.03 | 0.01 | 0.01 | 11 | 25 | 0.00 |
| 25 | 10 | 0.12 | 0.01 | 0.01 | 19 | 100 | 0.00 |
| 25 | 20 | 0.38 | 0.03 | 0.02 | 33 | 400 | 0.01 |
| 25 | 50 | 2.20 | 0.16 | 0.06 | 110 | 2500 | 0.08 |
| 25 | 100 | 8.37 | 0.91 | 0.18 | 307 | 10000 | 0.64 |
| 25 | 200 | 46.51 | 6.38 | 0.56 | 1025 | 40000 | 5.49 |
| 50 | 5 | 0.07 | 0.02 | 0.02 | 9 | 25 | 0.00 |
| 50 | 10 | 0.26 | 0.04 | 0.03 | 19 | 100 | 0.00 |
| 50 | 20 | 1.01 | 0.05 | 0.04 | 23 | 400 | 0.00 |
| 50 | 50 | 6.57 | 0.19 | 0.11 | 61 | 2500 | 0.06 |
| 50 | 100 | 25.42 | 0.91 | 0.30 | 161 | 10000 | 0.52 |
| 50 | 200 | 129.56 | 5.25 | 0.87 | 476 | 40000 | 4.03 |
| 100 | 5 | 0.45 | 0.15 | 0.12 | 9 | 25 | 0.00 |
| 100 | 10 | 1.68 | 0.24 | 0.22 | 17 | 100 | 0.00 |
| 100 | 20 | 6.59 | 0.28 | 0.25 | 19 | 400 | 0.00 |
| 100 | 50 | 41.51 | 0.67 | 0.57 | 43 | 2500 | 0.06 |
| 100 | 100 | 201.97 | 1.76 | 1.28 | 94 | 10000 | 0.37 |
| 100 | 200 | 981.37 | 6.52 | 3.43 | 247 | 40000 | 2.73 |
| 200 | 5 | 2.40 | 0.66 | 0.48 | 9 | 25 | 0.00 |
| 200 | 10 | 8.77 | 1.02 | 0.86 | 16 | 100 | 0.00 |
| 200 | 20 | 33.70 | 1.23 | 1.07 | 20 | 400 | 0.01 |
| 200 | 50 | 208.51 | 1.88 | 1.67 | 30 | 2500 | 0.04 |
| 200 | 100 | 856.82 | 3.67 | 3.17 | 58 | 10000 | 0.26 |
| 200 | 200 | 4143.22 | 10.08 | 7.52 | 139 | 40000 | 2.07 |

## 5.9. Συνάρτηση check_precision σε Πίνακες Τυχαίων Αριθμών για Εύρεση Ικανής Ακρίβειας

Επειδή **η διάσταση ενός πίνακα είναι δεδομένη** (από το πρόβλημα που έχουμε να λύσουμε), έχει σημασία **να βρούμε την ελάχιστη ακρίβεια η οποία είναι ικανοποιητική**, μας δίνει δηλαδή μια καλή εικόνα του ψευδοφάσματος, συναρτήσει πάντα της διάστασης του πίνακα.

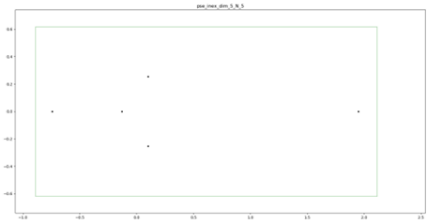
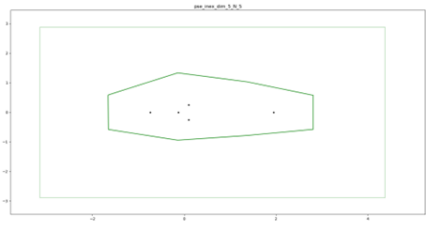
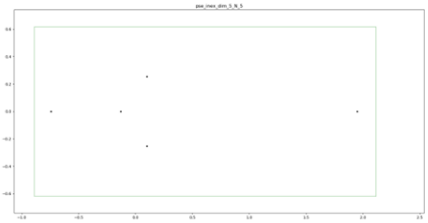
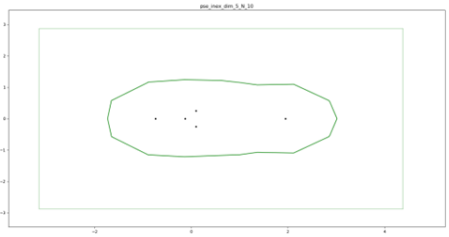
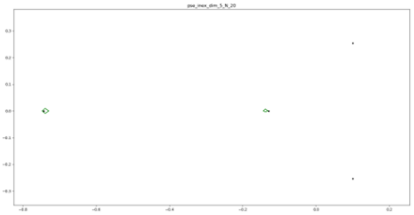
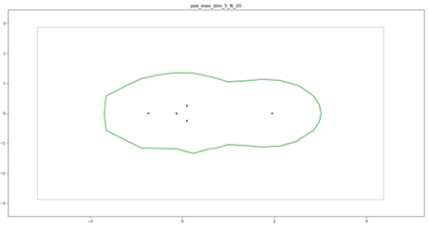
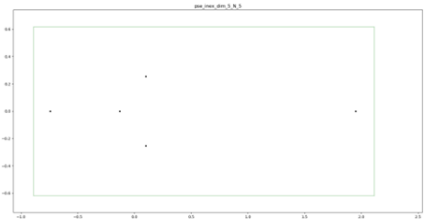
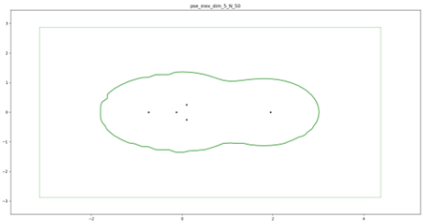
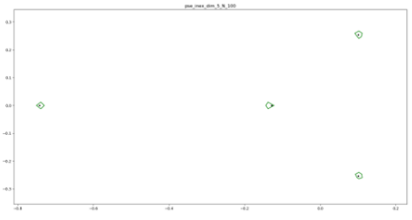
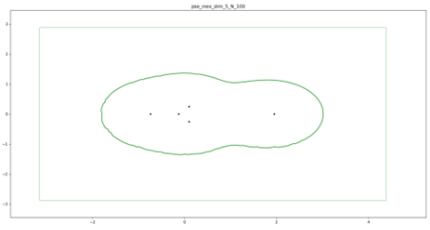
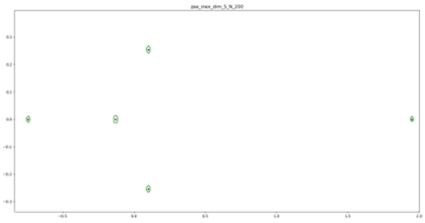
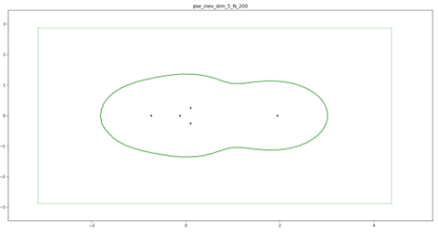
Δοκιμάζουμε ε = 0.01και ε = 1, για διαστάσεις από 5 x 5 έως και 200 x 200, και ακρίβειες από 5 x 5 έως και 200 x 200.

Από τα σχήματα των επόμενων σελίδων, συμπεραίνουμε τα εξής (για πίνακες τυχαίων αριθμών):

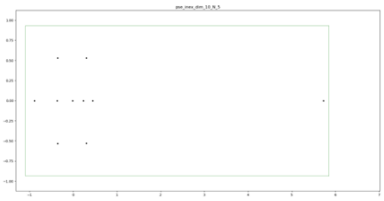
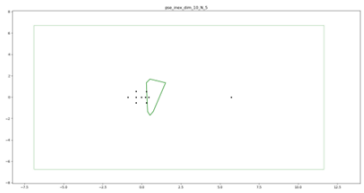
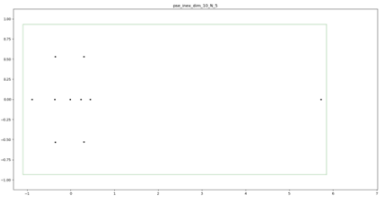
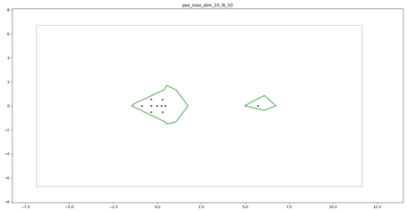
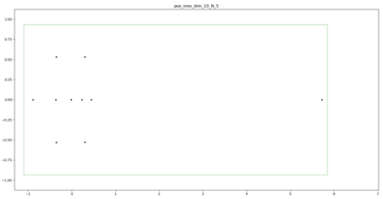
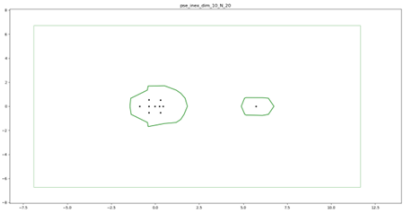
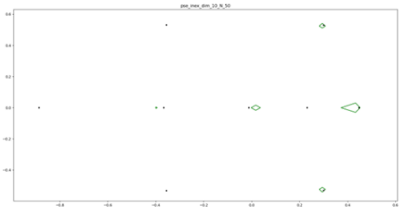
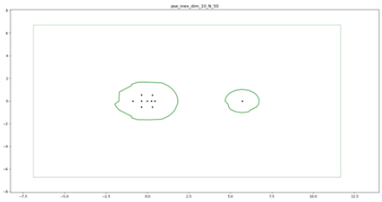
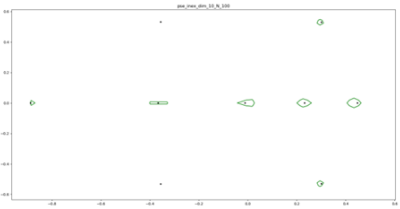
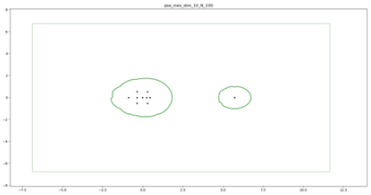
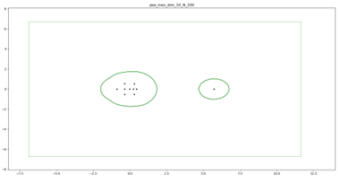
- Για ε = 1, ακρίβεια ίση με τη διάσταση δίνει γενικά άριστα αποτελέσματα.
- Για ε = 1, για διαστάσεις έως και 200 x 200, ακρίβεια 100 x100 είναι σχεδόν άψογη,
- Για ε =0.01, έως και 10 x 10, ακρίβεια 200 x 200 δίνει άριστα αποτελέσματα.
- Για ε =0.01, για διαστάσεις πάνω από 10 x 10, χρειάζεται ακρίβεια πάνω από 200 x 200

Κλήση: check_precision("inex", "random")

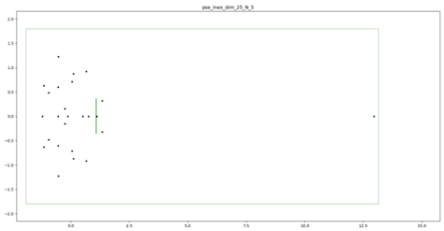
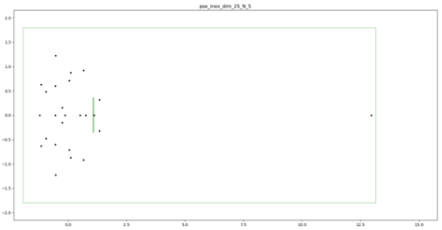
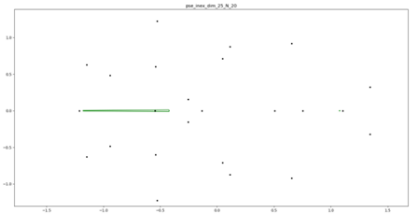
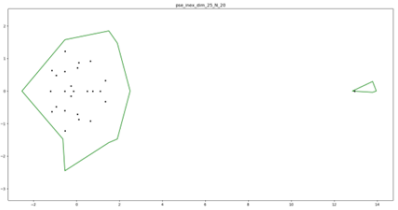
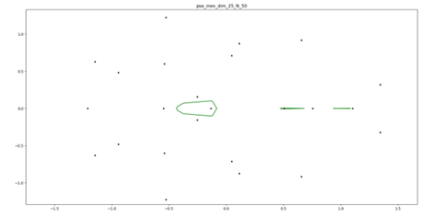
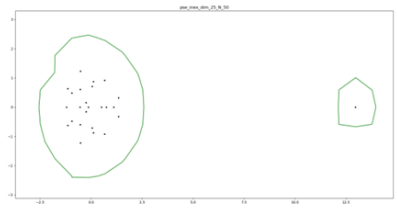
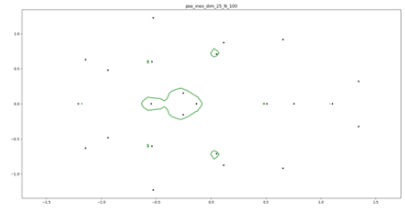
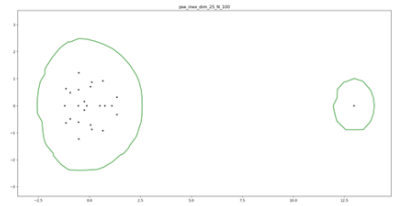
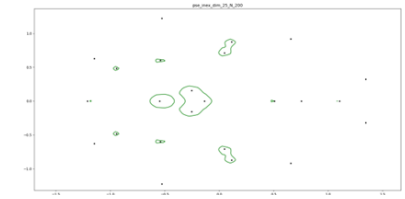
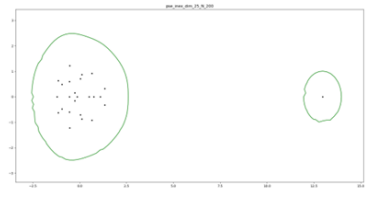
**dim = 5**

| | **ε = 0.01** | **ε = 1** |
|---|---|---|
| **N = 5** |  |  |
| **N = 10** |  |  |
| **N = 20**<br>στο ε=0.01 λεπτομέρεια |  |  |
| **N = 50**<br>στο ε=0.01 λεπτομέρεια |  |  |
| **N = 100**<br>στο ε=0.01 λεπτομέρεια |  |  |
| **N = 200**<br>στο ε=0.01 λεπτομέρεια |  |  |

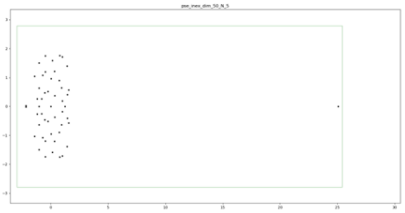
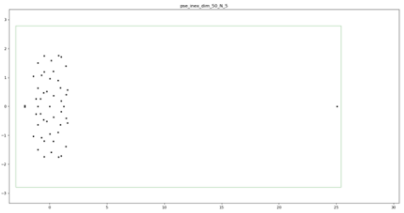
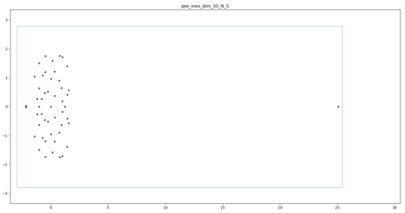
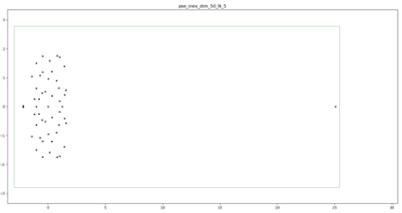
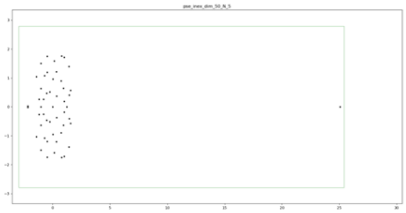
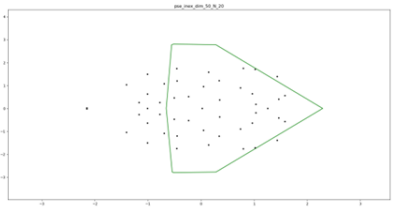
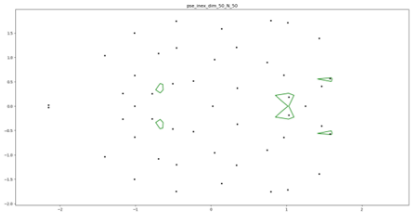
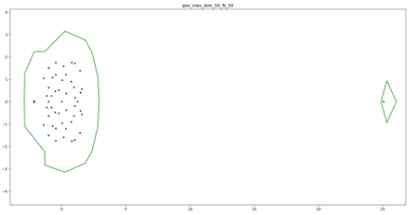
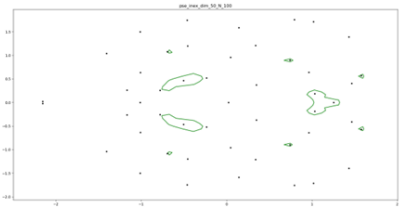
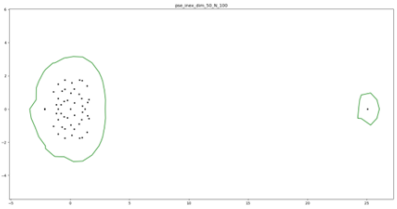
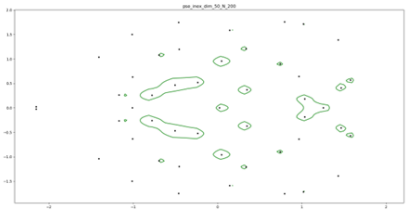
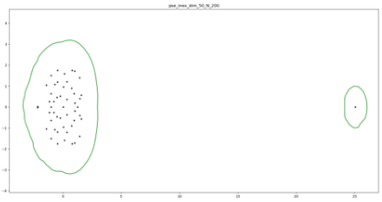
**dim = 10**

| | **ε = 0.01** | **ε = 1** |
|---|---|---|
| **N = 5** |  |  |
| **N = 10** |  |  |
| **N = 20** |  |  |
| **N = 50**<br>στο ε=0.01 λεπτομέρεια |  |  |
| **N = 100**<br>στο ε=0.01 λεπτομέρεια |  |  |
| **N = 200**<br>στο ε=0.01 λεπτομέρεια |  |  |

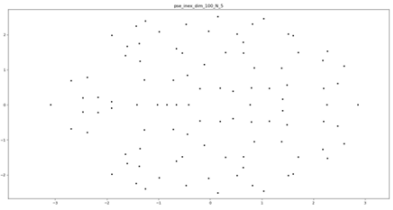
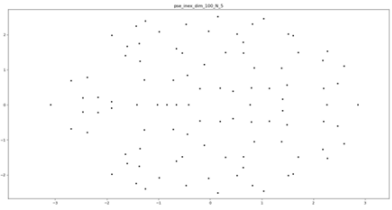
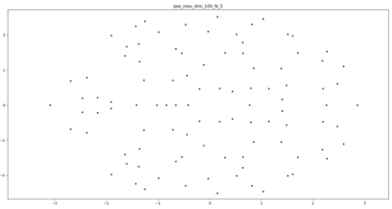
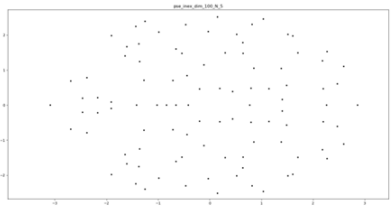
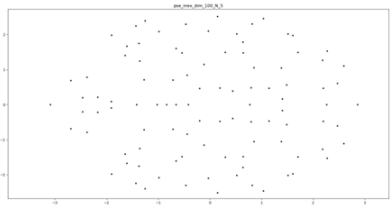
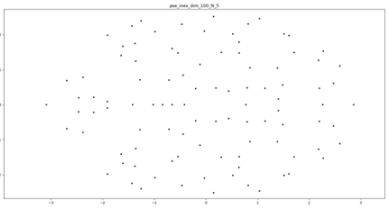
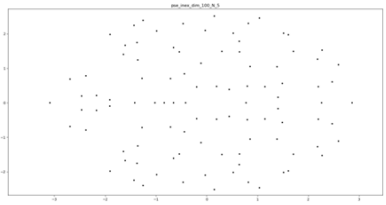
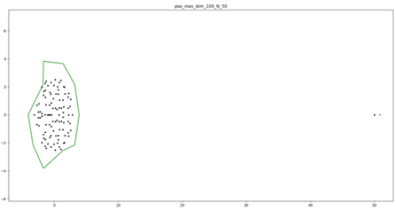
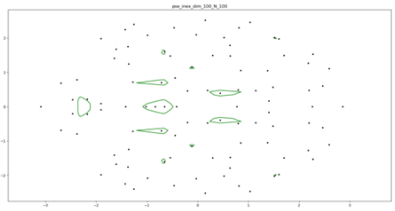
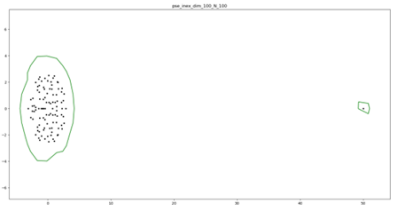
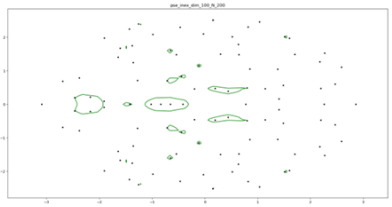
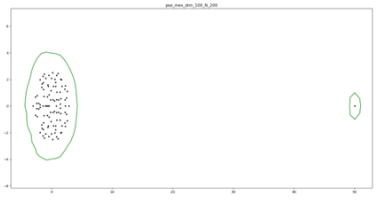
dim = 25

| | ε = 0.01 | ε = 1 |
|---|---|---|
| N = 5 |  |  |
| N = 10 |  |  |
| N = 20 |  |  |
| N = 50 |  |  |
| N = 100 |  |  |
| N = 200 |  |  |

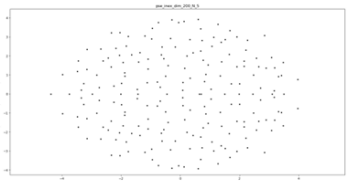
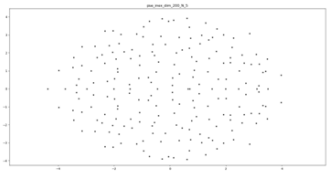
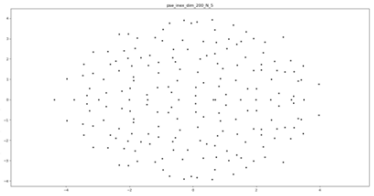
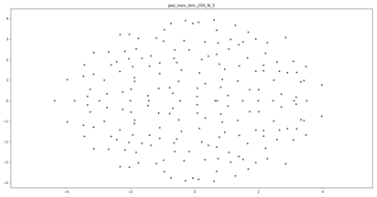
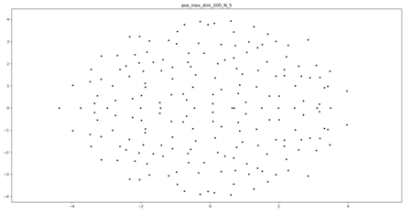
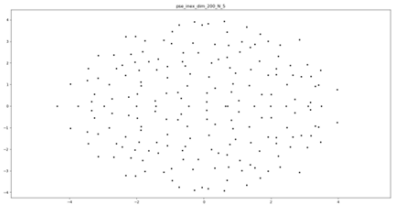
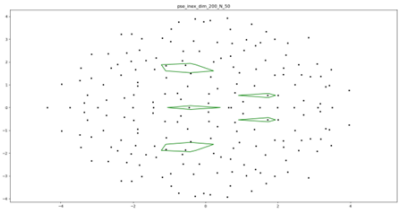
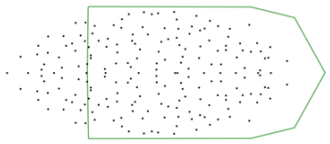
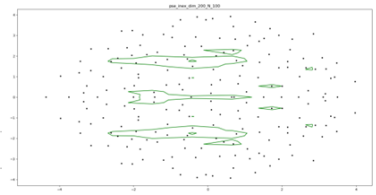
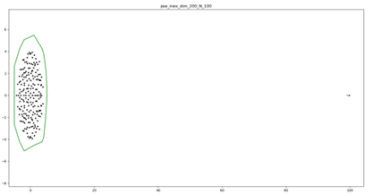
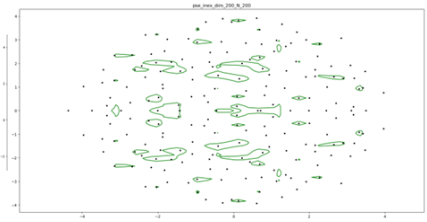
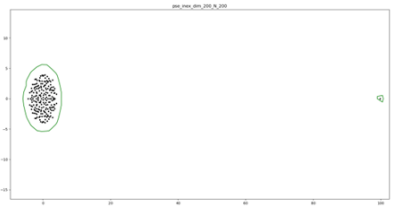
dim = 50

| | ε = 0.01 | ε = 1 |
|---|---|---|
| N = 5 |  |  |
| N = 10 |  |  |
| N = 20 |  |  |
| N = 50 |  |  |
| N = 100 |  |  |
| N = 200 |  |  |

dim = 100

| | ε = 0.01 | ε = 1 |
|---|---|---|
| N = 5 |  |  |
| N = 10 |  |  |
| N = 20 |  |  |
| N = 50 |  |  |
| N = 100 |  |  |
| N = 200 |  |  |

dim = 200

| | ε = 0.01 | ε = 1 |
|---|---|---|
| N = 5 |  |  |
| N = 10 |  |  |
| N = 20 |  |  |
| N = 50 |  |  |
| N = 100 |  |  |
| N = 200 |  |  |

## 5.10. Πίνακες Kahan

Θα κατασκευάσουμε πίνακες Kahan για διαστάσεις από 5 x 5 ως 200 x 200.

Θα υπολογίσουμε και θα απεικονίσουμε σε γράφημα το ψευδοφάσμα του Άγια διαφορετικές τιμές του ε (0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02) με ακρίβεια 300καθώς και το χωρίο Ω που κατασκεύασε ο αλγόριθμος για κάθε περίπτωση.

Συμπεράσματα:

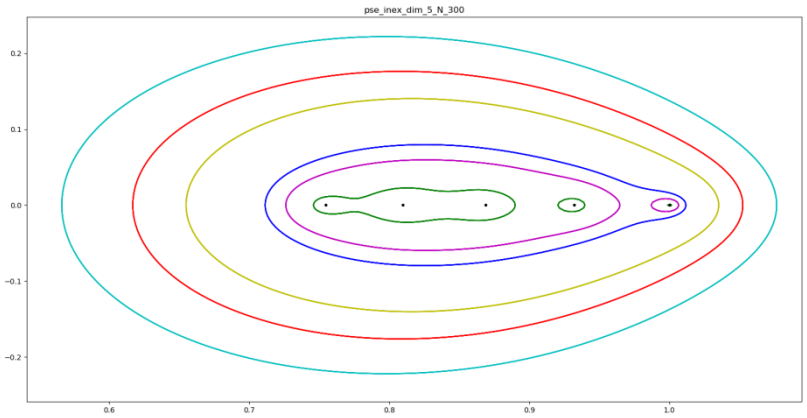
- Σε όλες τις περιπτώσεις το χωρίο Ω περιλάμβανε ολόκληρο το ψευδοφάσμα (ακόμη και στα γραφήματα που έχουμε κάνει μεγέθυνση και δε φαίνεται).
- Ο υπολογισμός του ψευδοφάσματος με τις παραμέτρους που θέσαμε ήταν ακριβής.
- Οι χρόνοι ήταν παρόμοιοι με αυτούς των τυχαίων πινάκων

### Κλήση συνάρτησης

```
check_pseudospectra(algorithm = "inex", data = "kahan")
```

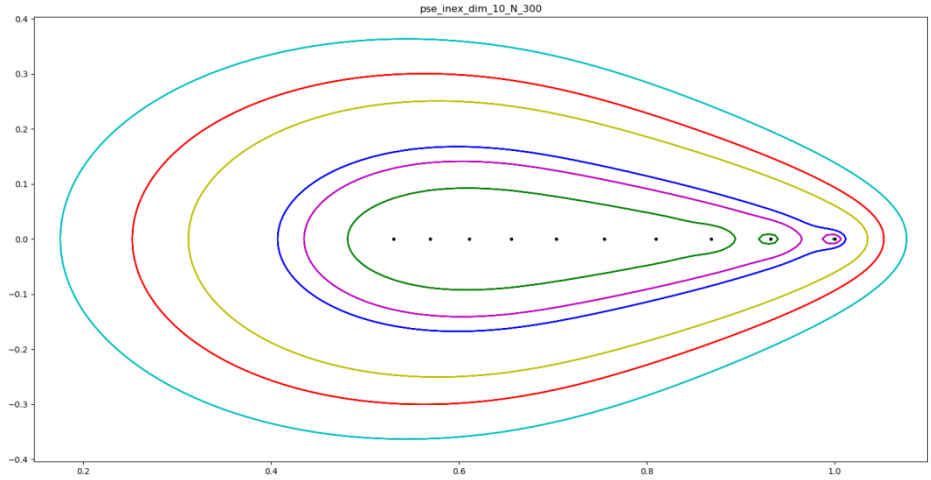
dim = 5, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



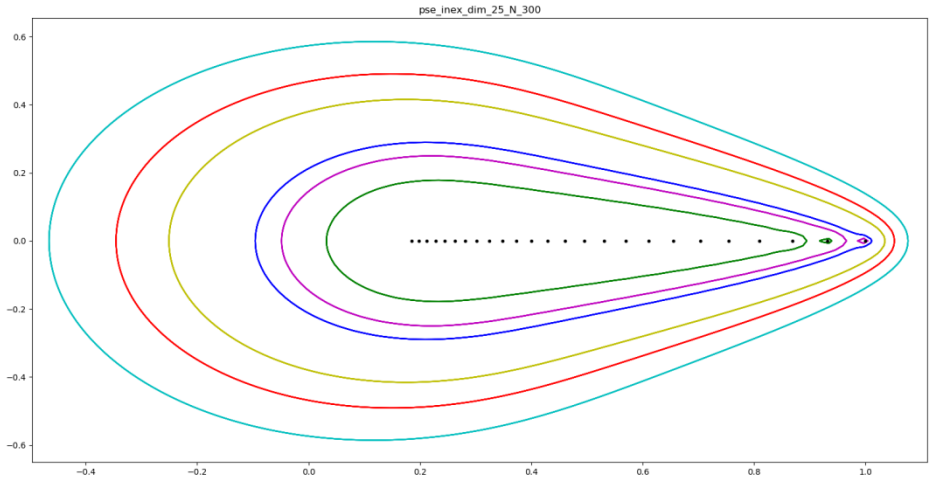

dim = 10, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



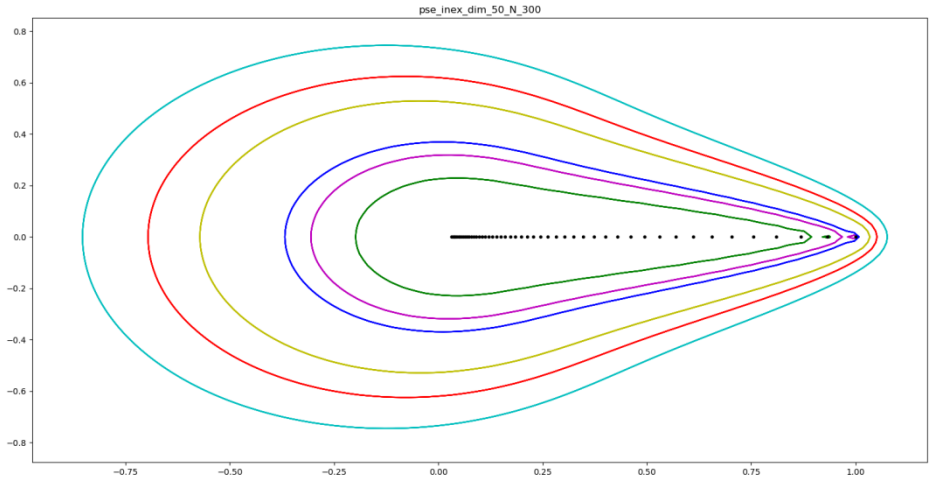

dim = 25, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



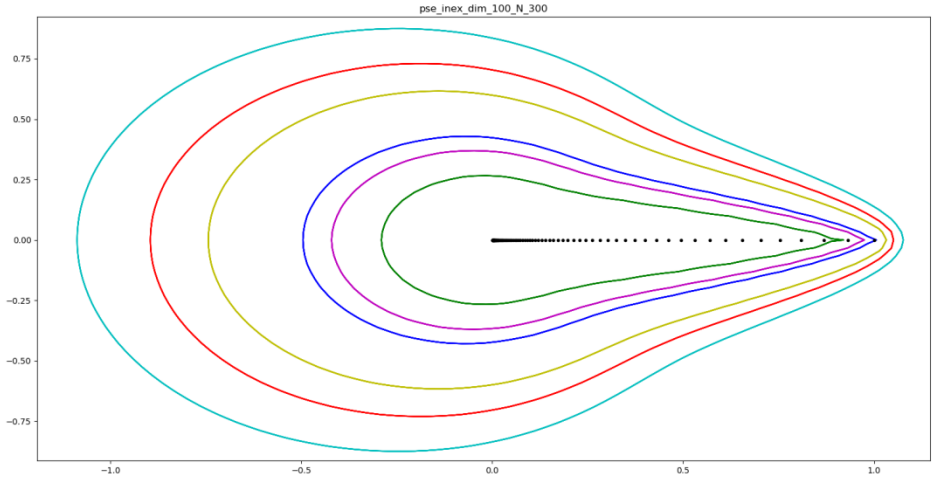
dim = 50, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



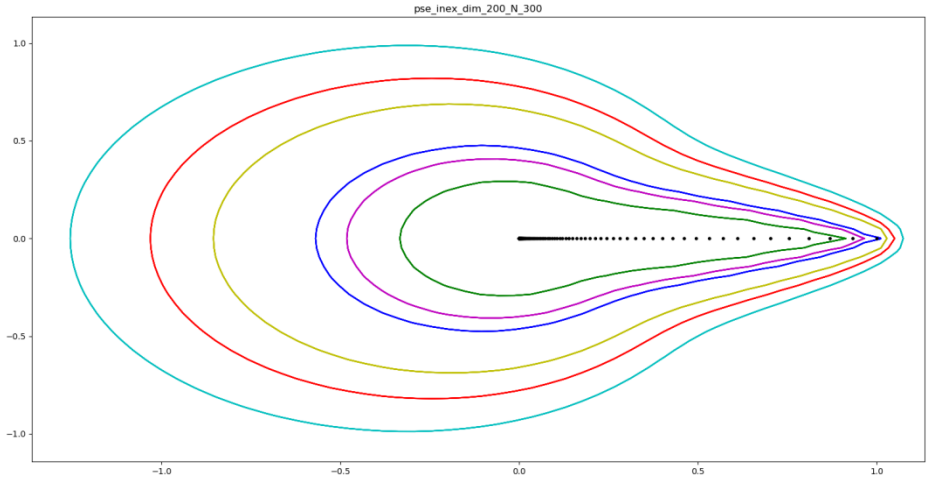

dim = 100, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02




dim = 200, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



Χρονομέτρηση

Κλήση συνάρτησης

```
check_time("inex", "kahan")
```

| dim | N | grid time | inex time | svd time | svds made | total svds N x N | exclusion time |
|---|---|---|---|---|---|---|---|
| 2 | 5 | 0.00 | 0.00 | 0.00 | 23 | 25 | 0.00 |
| 2 | 10 | 0.01 | 0.01 | 0.01 | 80 | 100 | 0.00 |
| 2 | 20 | 0.03 | 0.06 | 0.02 | 296 | 400 | 0.03 |
| 2 | 50 | 0.18 | 0.50 | 0.10 | 1623 | 2500 | 0.34 |
| 2 | 100 | 0.72 | 3.48 | 0.41 | 6200 | 10000 | 2.96 |
| 2 | 200 | 5.25 | 35.55 | 2.25 | 24161 | 40000 | 32.80 |
| 5 | 5 | 0.00 | 0.00 | 0.00 | 22 | 25 | 0.00 |
| 5 | 10 | 0.01 | 0.02 | 0.01 | 76 | 100 | 0.00 |
| 5 | 20 | 0.04 | 0.07 | 0.03 | 263 | 400 | 0.03 |
| 5 | 50 | 0.23 | 0.72 | 0.18 | 1438 | 2500 | 0.51 |
| 5 | 100 | 0.91 | 5.04 | 0.65 | 5442 | 10000 | 4.23 |
| 5 | 200 | 6.79 | 38.48 | 2.52 | 21127 | 40000 | 35.33 |
| 10 | 5 | 0.00 | 0.01 | 0.00 | 20 | 25 | 0.00 |
| 10 | 10 | 0.02 | 0.02 | 0.01 | 59 | 100 | 0.00 |
| 10 | 20 | 0.06 | 0.08 | 0.04 | 186 | 400 | 0.03 |
| 10 | 50 | 0.37 | 0.89 | 0.26 | 974 | 2500 | 0.58 |
| 10 | 100 | 1.47 | 4.50 | 0.79 | 3569 | 10000 | 3.55 |
| 10 | 200 | 10.19 | 31.66 | 2.89 | 13687 | 40000 | 28.13 |
| 25 | 5 | 0.02 | 0.02 | 0.02 | 13 | 25 | 0.00 |
| 25 | 10 | 0.06 | 0.05 | 0.04 | 39 | 100 | 0.00 |
| 25 | 20 | 0.22 | 0.15 | 0.11 | 102 | 400 | 0.03 |
| 25 | 50 | 1.38 | 0.75 | 0.44 | 463 | 2500 | 0.27 |
| 25 | 100 | 6.12 | 3.81 | 1.43 | 1606 | 10000 | 2.23 |
| 25 | 200 | 39.26 | 21.34 | 4.61 | 5942 | 40000 | 16.08 |
| 50 | 5 | 0.07 | 0.04 | 0.03 | 12 | 25 | 0.00 |
| 50 | 10 | 0.23 | 0.08 | 0.07 | 28 | 100 | 0.00 |
| 50 | 20 | 1.12 | 0.21 | 0.19 | 75 | 400 | 0.01 |
| 50 | 50 | 6.43 | 0.94 | 0.69 | 299 | 2500 | 0.21 |
| 50 | 100 | 24.23 | 4.04 | 2.24 | 971 | 10000 | 1.68 |
| 50 | 200 | 125.94 | 21.14 | 7.98 | 3453 | 40000 | 12.68 |
| 100 | 5 | 0.43 | 0.24 | 0.21 | 12 | 25 | 0.00 |
| 100 | 10 | 1.64 | 0.43 | 0.40 | 24 | 100 | 0.00 |
| 100 | 20 | 6.69 | 1.12 | 1.07 | 64 | 400 | 0.02 |
| 100 | 50 | 47.40 | 4.07 | 3.80 | 220 | 2500 | 0.20 |
| 100 | 100 | 195.86 | 12.90 | 11.22 | 652 | 10000 | 1.53 |
| 100 | 200 | 947.11 | 51.04 | 37.51 | 2198 | 40000 | 13.03 |
| 200 | 5 | 1.99 | 0.90 | 0.77 | 12 | 25 | 0.00 |
| 200 | 10 | 7.34 | 1.62 | 1.49 | 23 | 100 | 0.00 |
| 200 | 20 | 30.54 | 3.75 | 3.60 | 54 | 400 | 0.02 |
| 200 | 50 | 219.05 | 11.61 | 11.23 | 171 | 2500 | 0.22 |
| 200 | 100 | 828.37 | 33.85 | 31.86 | 482 | 10000 | 1.72 |
| 200 | 200 | 4048.26 | 133.08 | 118.40 | 1523 | 40000 | 14.06 |

## 5.11. Πίνακες Toeplitz GrCar

Θα κατασκευάσουμε πίνακες Toeplitz GrCar για διαστάσεις από 5 x 5 ως 100 x100.

Θα υπολογίσουμε και θα απεικονίσουμε σε γράφημα το ψευδοφάσμα του Λια διαφορετικές τιμές του ε (0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02) με ακρίβεια 300καθώς και το χωρίο Ω που κατασκεύασε ο αλγόριθμος για κάθε περίπτωση.

Συμπεράσματα:

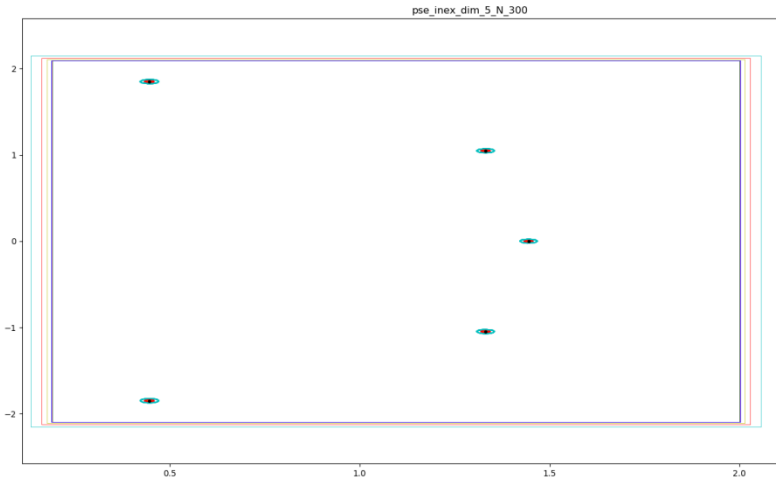
- Σε όλες τις περιπτώσεις το χωρίο Ω περιλάμβανε ολόκληρο το ψευδοφάσμα (ακόμη και στα γραφήματα που έχουμε κάνει μεγέθυνση και δε φαίνεται).
- Ο υπολογισμός του ψευδοφάσματος με τις παραμέτρους που θέσαμε ήταν ακριβής.
- Οι χρόνοι ήταν λίγο μικρότεροι από αυτούς των τυχαίων πινάκων

```
check_pseudospectra(algorithm = "inex", data = "grcar")
```

## Σύγκριση χωρίων και ψευδοφασμάτων

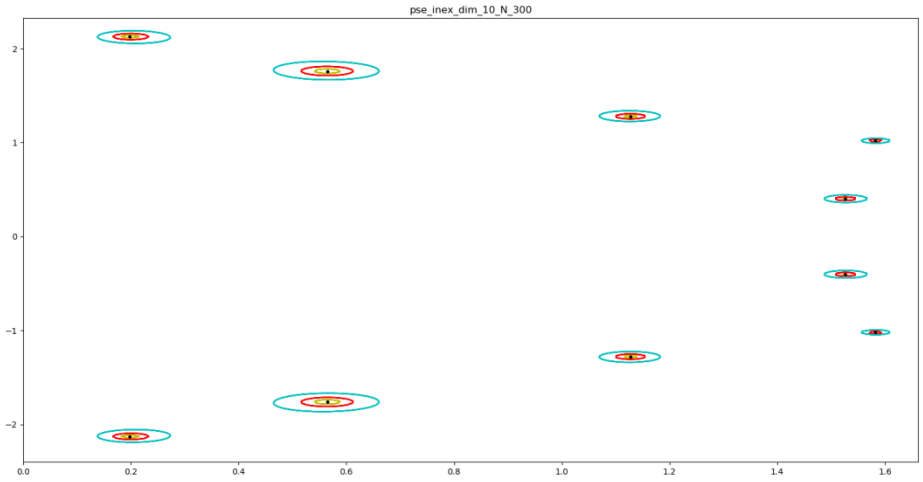
dim = 5, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



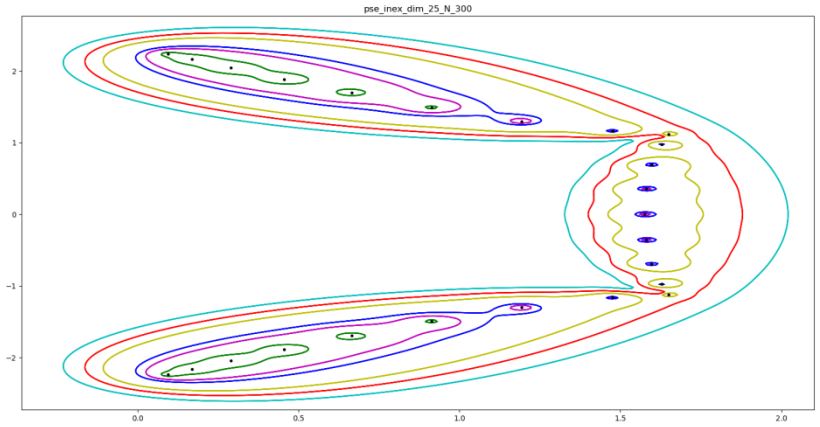

dim = 10, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



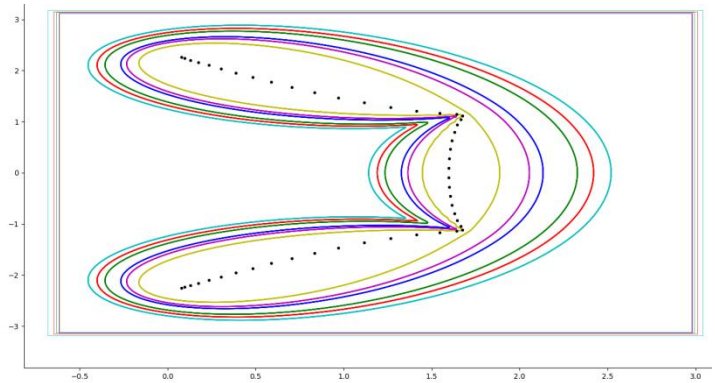
dim = 25, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



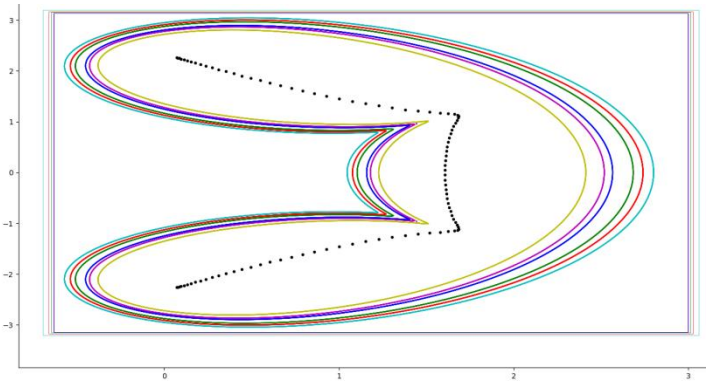

dim = 50, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



dim = 100, N = 300

ε = 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02



Χρονομέτρηση

Κλήσησυνάρτησης

```
check_time("inex", "grcar")
```

| dim | N | grid time | inex time | svd time | svds made | total svds N x N | exclusion time |
|---|---|---|---|---|---|---|---|
| 2 | 5 | 0.00 | 0.00 | 0.00 | 23 | 25 | 0.00 |
| 2 | 10 | 0.01 | 0.01 | 0.00 | 77 | 100 | 0.00 |
| 2 | 20 | 0.03 | 0.04 | 0.01 | 250 | 400 | 0.02 |
| 2 | 50 | 0.16 | 0.47 | 0.08 | 1313 | 2500 | 0.36 |
| 2 | 100 | 0.65 | 3.49 | 0.28 | 4955 | 10000 | 3.10 |
| 2 | 200 | 5.36 | 29.07 | 1.17 | 19141 | 40000 | 27.48 |
| 5 | 5 | 0.00 | 0.00 | 0.00 | 16 | 25 | 0.00 |
| 5 | 10 | 0.01 | 0.01 | 0.00 | 46 | 100 | 0.00 |
| 5 | 20 | 0.03 | 0.04 | 0.01 | 135 | 400 | 0.02 |
| 5 | 50 | 0.21 | 0.44 | 0.06 | 641 | 2500 | 0.34 |
| 5 | 100 | 0.83 | 3.00 | 0.23 | 2268 | 10000 | 2.66 |
| 5 | 200 | 6.22 | 23.26 | 0.84 | 8462 | 40000 | 21.95 |
| 10 | 5 | 0.00 | 0.00 | 0.00 | 16 | 25 | 0.00 |
| 10 | 10 | 0.01 | 0.01 | 0.01 | 52 | 100 | 0.00 |
| 10 | 20 | 0.06 | 0.05 | 0.02 | 149 | 400 | 0.02 |
| 10 | 50 | 0.35 | 0.54 | 0.13 | 707 | 2500 | 0.37 |
| 10 | 100 | 1.64 | 3.81 | 0.50 | 2543 | 10000 | 3.18 |
| 10 | 200 | 10.25 | 28.14 | 1.77 | 9541 | 40000 | 25.87 |
| 25 | 5 | 0.02 | 0.01 | 0.01 | 16 | 25 | 0.00 |
| 25 | 10 | 0.07 | 0.05 | 0.04 | 53 | 100 | 0.00 |
| 25 | 20 | 0.29 | 0.15 | 0.12 | 158 | 400 | 0.03 |
| 25 | 50 | 1.80 | 0.96 | 0.57 | 792 | 2500 | 0.35 |
| 25 | 100 | 7.87 | 5.26 | 2.22 | 2913 | 10000 | 2.89 |
| 25 | 200 | 38.46 | 33.79 | 8.30 | 11134 | 40000 | 24.95 |
| 50 | 5 | 0.06 | 0.05 | 0.04 | 17 | 25 | 0.00 |
| 50 | 10 | 0.22 | 0.14 | 0.12 | 52 | 100 | 0.00 |
| 50 | 20 | 0.88 | 0.40 | 0.37 | 163 | 400 | 0.02 |
| 50 | 50 | 5.79 | 2.28 | 1.89 | 820 | 2500 | 0.34 |
| 50 | 100 | 21.52 | 9.93 | 7.04 | 3043 | 10000 | 2.73 |
| 50 | 200 | 122.01 | 51.63 | 28.18 | 11640 | 40000 | 22.75 |
| 100 | 5 | 0.36 | 0.28 | 0.24 | 17 | 25 | 0.00 |
| 100 | 10 | 1.44 | 0.86 | 0.83 | 53 | 100 | 0.00 |
| 100 | 20 | 5.88 | 2.49 | 2.43 | 164 | 400 | 0.02 |
| 100 | 50 | 35.94 | 12.95 | 12.54 | 828 | 2500 | 0.33 |
| 100 | 100 | 156.65 | 45.01 | 42.24 | 3080 | 10000 | 2.54 |
| 100 | 200 | 743.23 | 191.33 | 169.54 | 11820 | 40000 | 21.05 |
| 200 | 5 | 2.58 | 1.50 | 1.27 | 17 | 25 | 0.00 |
| 200 | 10 | 8.48 | 4.29 | 4.05 | 53 | 100 | 0.00 |
| 200 | 20 | 30.78 | 12.68 | 12.44 | 164 | 400 | 0.03 |
| 200 | 50 | 193.62 | 60.11 | 59.52 | 829 | 2500 | 0.31 |
| 200 | 100 | 865.37 | 247.51 | 244.25 | 3095 | 10000 | 2.85 |
| 200 | 200 | 3643.28 | 999.08 | 974.01 | 11871 | 40000 | 23.91 |

# 6. Συμπεράσματα και Μελλοντικές Προοπτικές

## 6.1. Συμπεράσματα από τους Αλγόριθμους GRID και Inclusion-Exclusion

Οι **αλγόριθμοι Grid** επιλέγουν ένα **πλέγμα** του μιγαδικού επιπέδου και αποφαίνονται αν τα σημεία του ανήκουν στο ψευδοφάσμα εξετάζοντάς τα. Οι αλγόριθμοι Grid βγάζουν σωστά αποτελέσματα. Το πλέγμα από το οποίο θα ξεκινήσουμε πρέπει να περιέχει το ψευδοφάσμα και ταυτόχρονα να το προσεγγίζει σε ικανοποιητικό βαθμό έτσι ώστε να μην κάνουμε αχρείαστα μεγάλο αριθμό υπολογισμών. Αυτό είναι αρκετά δύσκολο και συνήθως το αρχικό πλέγμα είναι πολύ μεγαλύτερο από το ψευδοφάσμα.

Ένα βασικό ερώτημα είναι ποια σημεία του μιγαδικού επιπέδου θα εξεταστούν. **Ο αριθμός των εξεταζόμενων σημείων αποτελεί την κύρια παράμετρο αύξησης του χρονικού κόστους του αλγορίθμου.**

Ο αλγόριθμος GRID είναι απλός στην εφαρμογή και προσφέρει παραλληλισμό σε πολλά επίπεδα. Επιτυγχάνεται παραλληλισμός αφού οι υπολογισμοί της μοναδικής τιμής σε κάθε σημείο είναι ανεξάρτητοι. Ο παραλληλισμός μπορεί να επιτευχθεί κατά τον υπολογισμό κάθε μικρότερης μοναδικής τιμής.

Παρά τη δυνατότητα για σχεδόν ιδανική επιτάχυνση σε ένα παράλληλο σύστημα, ο αλγόριθμος GRID συνεπάγεται πολύ περιττούς υπολογισμούς. Συγκεκριμένα, η μικρότερη μοναδική τιμή υπολογίζεται σε διαφορετικούς πίνακες, αλλά όλοι είναι μετατοπίσεις ενός και του αυτού πίνακα. Για τον χειρισμό μεγάλων πινάκων, ο αλγόριθμος προτείνει την ανάμειξη δύο προσεγγίσεων: μείωση πυκνού πίνακα και μείωση διαστάσεων στον τομέα.

Ο αλγόριθμος μπορεί να υλοποιηθεί με υψηλή ταχύτητα με απλή στατική εκχώρηση υπολογισμών ανά επεξεργαστή, δεδομένου ότι το κόστος υπολογισμού της μικρότερης μοναδικής τιμής είναι τουλάχιστον τετραγωνικό για κάθε σημείο του πλέγματος. Η εξισορρόπηση φορτίου γίνεται επίσης ζήτημα όταν το κόστος υπολογισμού της μικρότερης μοναδικής τιμής ποικίλλει πολύ στον τομέα.

Στην αρχική εκδοχή του αλγόριθμου **Inclusion-Exclusion**, το exclusion γινόταν με ένα διπλό loop που έτρεχε κάθε φορά που γινόταν κάποιο exclusion, ώστε να βρεθούν τα στοιχεία του πίνακα Z που πρέπει να εξαχθούν από την εξέταση στη συνέχεια. Αυτό οδηγεί σε πολυπλοκότητα χρόνου exclusion $O(N^2)$ όπου N είναι η ακρίβεια (πλήθος σημείων ανά άξονα). Βελτιώσαμε το exclusion βάζοντας τα στοιχεία του Z μαζί με τα indices σε μια μονοδιάστατη λίστα. Κάθε φορά που γίνεται κάποιο exclusion, βγάζουμε τα τα στοιχεία του πίνακα Z που πρέπει να εξαχθούν από την εξέταση από τη λίστα, ώστε να μην τα εξετάσουμε στη συνέχεια. Αυτό οδηγεί σε πολυπλοκότητα χρόνου exclusion O(N). Το exclusion time με τον βελτιωμένο τρόπο αυξάνεται γραμμικά.

## 6.2. ΟΑλγόριθμος Path-Following του Bruhl

Οι μέθοδοι ιχνηλάτησης καμπύλης (path-following) αντιπροσωπεύουν μια κρίσιμη κατηγορία τεχνικών που στοχεύουν στην χάραξη καμπυλών επιλεγμένων ισοϋψών. Αυτές οι μέθοδοι αξιοποιούν έννοιες από ομοτοπίες και προβλήματα ιχνηλάτησης καμπυλών, που απαντώνται συνήθως στην αριθμητική επίλυση προβλημάτων αρχικής τιμής και δυναμικών συστημάτων.

Η πρόταση για τη χρήση αυτών των τεχνικών για τον υπολογισμό ψευδοφασμάτων έγινε για πρώτη φορά από τον Kostin, με τον πρώτο σχετικό αλγόριθμο να πιστώνεται στον M. Brühl [BRUHL1996]. Οι μεταγενέστερες εξελίξεις οδήγησαν στην εισαγωγή των μεθόδων παράλληλης ανίχνευσης καμπυλών από τους Κ. Μπέκα και Ε. Γαλλόπουλο [BEKAS2001], [BEKAS2002]. Επιπλέον, προτάθηκαν αλγόριθμοι ανίχνευσης καμπυλών από τους B. Philippe, D. Mezher [MEZHER2002a], [MEZHER2002b].

Μια βασική διάκριση μεταξύ αυτών των μεθόδων ανίχνευσης καμπυλών και των προηγούμενων έγκειται στην προσέγγισή τους. Οι τελευταίες μέθοδοι χρησιμοποιούν ένα ομοιόμορφο τριγωνικό πλέγμα και εξαρτώνται από τις τεχνικές διχοτόμησης για τον εντοπισμό μιας ρίζας της συνάρτησης $\sigma_{min}(A-zI) -\varepsilon = 0$ σε ευθύγραμμα τμήματα μεταξύ των κόμβων του πλέγματος. Αντίθετα, οι προηγούμενες μέθοδοι χρησιμοποιούσαν τη μέθοδο Newton χωρίς καμία προκαθορισμένη γεωμετρία.

Ο αλγόριθμος **Path-Following** του Bruhl ξεκινά βρίσκοντας ένα σημείο του επιπέδου το οποίο ανήκει στην καμπύλη $\partial\Lambda_\varepsilon$ και στη συνέχεια υπολογίζει το επόμενό του κ.ο.κ, προβλέποντας κάθε φορά το επόμενο σημείο και διορθώνοντας την πρόβλεψη. Τελικά ο αλγόριθμος υπολογίζει ολόκληρο το σύνορο του ψευδοφάσματος έχοντας ακολουθήσει το μονοπάτι που ορίζουν οι υπολογισμοί των σημείων. Το πρώτο σημείο βρίσκεται συνήθως ξεκινώντας από μια τυχαία ιδιοτιμή του πίνακα, η οποία σίγουρα θα ανήκει στο εσωτερικό του ψευδοφάσματος, και ακολουθώντας μια συγκεκριμένη κατεύθυνση.

Η ιδέα του path-following μειώνει σημαντικά το χρόνο εκτέλεσης σε σχέση με τους αλγορίθμους Grid, αλλά έχει και αυτή ορισμένα βασικά μειονεκτήματα. Το μεγαλύτερο πρόβλημα παρουσιάζεται, όπως θα δούμε, όταν το ψευδοφάσμα δεν είναι συνεκτικό σύνολο, κάτι που συμβαίνει στις περισσότερες περιπτώσεις για μικρές τιμές του ε. Εκ φύσεως οι αλγόριθμοι που ανήκουν σε αυτή την κατηγορία θα υπολογίσουν μόνο μια συνεκτική συνιστώσα του ψευδοφάσματος και για τον υπολογισμό των υπολοίπων πρέπει να εκτελεστούν εκ νέου, με αφετηρία κάποια άλλη ιδιοτιμή. Ένα άλλο πρόβλημα είναι ότι καθώς ακολουθούν την καμπύλη, αν γίνει κάποιος λάθος υπολογισμός ή σε σημεία που αυτή έχει περίεργη γεωμετρία (όπως απότομες αλλαγές κατεύθυνσης ή μικρή απόσταση μεταξύ δύο συνεκτικών συνιστωσών) μπορεί να δώσουν λάθος αποτελέσματα. Το πρόβλημα αυτό λύνεται προβλέποντας σημεία πολύ κοντά στα ήδη υπάρχοντα, γεγονός που όμως όπως είναι φανερό οδηγεί στην ανάγκη υπολογισμού περισσότερων σημείων και άρα στην αύξηση του χρόνου εκτέλεσης.

## 6.3. Περιγραφή του Αλγορίθμου του Bruhl σε Ψευδοκώδικα

ΕΙΣΟΔΟΣ:

$\mathbf{A} \in \mathbb{C}^{n \times n}$, $\varepsilon > 0$

$K$: ο αριθμός των σημείων του συνόρου που θα υπολογιστούν

$\mathbf{d}_0$ : η διεύθυνση για το πρώτο σημείο, συνήθως $\mathbf{d}_0 = 1 + i$

$tol$: η ακρίβεια για το πρώτο σημείο, συνήθως $10^{-6}$

$\tau$: το βήμα για την πρόβλεψη, συνήθως $10^{-3}$

ΒΗΜΑ 0: Υπολογισμός του πρώτου σημείου

$z_1^{new} = \lambda_0 + \frac{\varepsilon}{2}\mathbf{d}_0$ , όπου $\lambda_0$ για τυχαία ιδιοτιμή του πίνακα.

**Όσο** $\left| z_1^{new} - z_1^{old} \right| > tol$

$z_1^{old} = z_1^{new}$

Υπολογισμός της τριπλέτας $\left(s_{\min}, \mathbf{u}_{\min}, \mathbf{v}_{\min}\right)$ με SVD για τον πίνακα $z_1^{old}\mathbf{I} - \mathbf{A}$

Υπολογισμός του $z_1^{new}$ , σύμφωνα με την σχέση (5.2)

**Τέλος_επανάληψης**

$z_1 = z_1^{new}$

**Για** $k = 2, 3, \ldots, K$

ΒΗΜΑ 1: Πρόβλεψη

Υπολογισμός της τριπλέτας $\left(s_{\min}, \mathbf{u}_{\min}, \mathbf{v}_{\min}\right)$ με SVD για τον πίνακα $z_{k-1}\mathbf{I} - \mathbf{A}$

$\mathbf{r}_k = i\frac{\mathbf{v}_{\min}^{*}\mathbf{u}_{\min}}{\left|\mathbf{v}_{\min}^{*}\mathbf{u}_{\min}\right|}$

$\tilde{z}_k = z_{k-1} + \tau\mathbf{r}_k$

ΒΗΜΑ 2: Διόρθωση

Υπολογισμός του $z_k$ , σύμφωνα με την σχέση (5.3), αντικαθιστώντας όπου λ το $\tilde{z}_k$ και κάνοντας χρήση της τριπλέτας για το $z_{k-1}$.

**Τέλος_επανάληψης**

**Τέλος.**

## 6.4. MatlabΚώδικας για τον Αλγόριθμο του Bruhl

**Αρχείο bruhl_01_program.m**

```
A = [    0.0760 0.4173 0.4893 0.7803 0.1320
         0.2399 0.0497 0.3377 0.3897 0.9421
         0.1233 0.9027 0.9001 0.2417 0.9561
         0.1839 0.9448 0.3692 0.4039 0.5752
         0.2400 0.4909 0.1112 0.0965 0.0598
    ];

epsilons = [0.8, 0.5, 0.2, 0.1];
K = 50000 ;
d0 = complex(1,1) ;
tol = 10^-6 ;
t = 10^-3 ;

for i = 1:length(epsilons)
    epsilon = epsilons(i);
fprintf("\n calling bruhl with epsilon = %.2f \n", epsilon);
bruhl(A, epsilon, K, d0, tol, t, 0, 'random.jpg');
end
```

**Αρχείο bruhl.m**

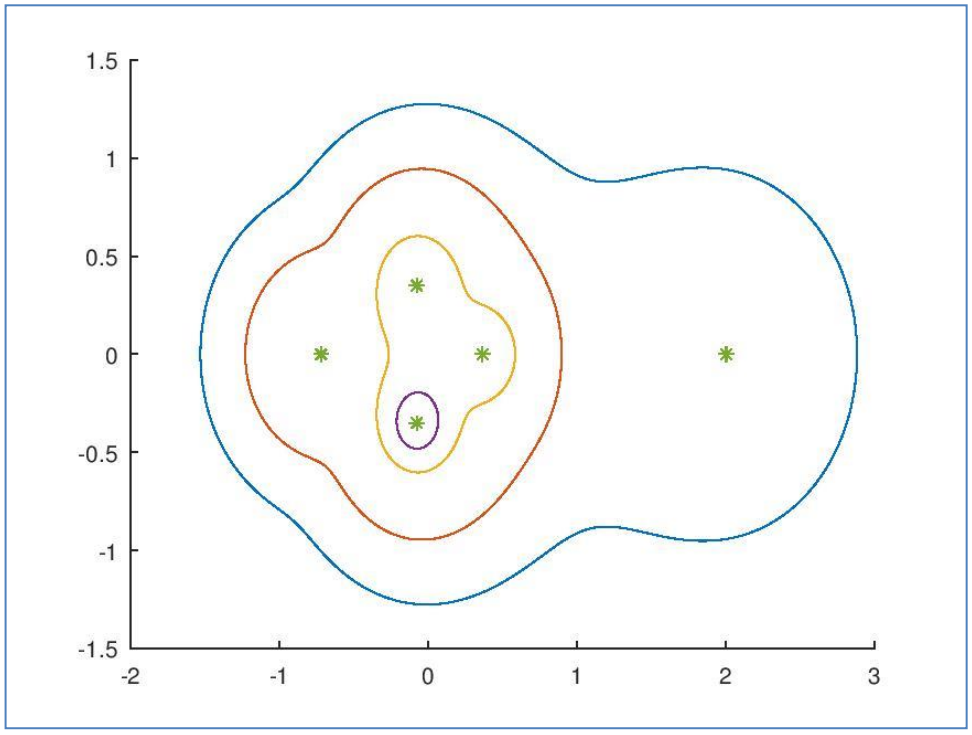
```
function z = bruhl(A, epsilon, K, d0, tol, t, save, name)
figure(1);   %% create a figure
    hold on;
    n = length(A);
    eigs = sort(eig(A));
disp("\n    eigs \n");
    disp(eigs);
    l0 = eigs(1);
disp("\n    L0 \n");
    disp(l0);
    s = svd(A);
disp("\n   s \n");
    disp(s);
    z = zeros(K,1);

    %% Find the first point
    z1old = l0 + (epsilon / 2) * d0;
    [U, S, V] = svd(z1old  *eye(n) - A);
    umin = U(:, end);
    smin = S(end);
    vmin = V(:, end);
    z1new = z1old - ( (smin - epsilon) / real( (conj(d0) * (vmin') * umin) ) ) * d0;
    count=1;

    while (abs (z1new - z1old) > tol)
        z1old = z1new;
        [U, S, V] = svd(z1old * eye(n) - A);
        umin = U(:, end);
        smin = S(end);
        vmin = V(:, end);
        z1new = z1old - ( (smin - epsilon) / real( (conj(d0) * (vmin') * umin) ) ) * d0;
        count = count + 1;
    end
```

```
z(1) = z1new;

    %For the rest of the points
    for (k = 2 : K)
        %% Prediction
        [U,S,V] = svd(z(k-1)*eye(n)-A); %the triplet for z(k-1)
        umin = U(:,end);
        smin = S(end);
        vmin = V(:,end);
        grad = (vmin')*umin;
        %% disp(grad);
        r = (grad*i)/abs(grad);
%% disp(r);
        pred = z(k-1)+t*r;
%% disp(pred);
        %% Correction
        z(k) = pred-((smin-epsilon)/((umin')*vmin));
        %% disp(z(k));
        %% disp("\n\n");
    end
    %% plot(z, 'g');
    plot(z);
    if (save == 1)
plot(eigs, 'linestyle', 'none', 'marker', '*');
print(name, "-djpg") ;
    end
    %% close(1);
    %% end
    %% disp(count);
end
```

## 6.5. Αποτελέσματα του Κώδικα Matlab

```
calling bruhl with epsilon = 0.80

    eigs

  -0.0783 - 0.3508i
  -0.0783 + 0.3508i
   0.3640 +      0i
  -0.7208 +      0i
   2.0028 +      0i

    L0

-0.078255 - 0.350776i

   s

   2.3595
   0.7814
   0.6429
   0.3706
   0.1545

 calling bruhl with epsilon = 0.50

    eigs

  -0.0783 - 0.3508i
  -0.0783 + 0.3508i
   0.3640 +      0i
  -0.7208 +      0i
   2.0028 +      0i

    L0

-0.078255 - 0.350776i

   s

   2.3595
   0.7814
   0.6429
   0.3706
   0.1545
```

```
 calling bruhl with epsilon = 0.20

    eigs

  -0.0783 - 0.3508i
  -0.0783 + 0.3508i
   0.3640 +       0i
  -0.7208 +       0i
   2.0028 +       0i

    L0

-0.078255 - 0.350776i

    s

   2.3595
   0.7814
   0.6429
   0.3706
   0.1545

 calling bruhl with epsilon = 0.10

    eigs

  -0.0783 - 0.3508i
-0.0783 + 0.3508i
   0.3640 +       0i
  -0.7208 +       0i
   2.0028 +       0i

    L0

-0.078255 - 0.350776i

    s

   2.3595
   0.7814
   0.6429
   0.3706
   0.1545
```



## 6.6. Python Κώδικας για τον Αλγόριθμο του Bruhl

**Αρχείο bruhl.py**

```
import numpy as np
from scipy.linalg import eigvals
from numpy.linalg import svd
import matplotlib.pyplot as plt

import sys
def printf(format, *args):
sys.stdout.write(format % args)

def bruhl(A, epsilon, K, d0, tol, t, save, name):
plt.figure()
    n = len(A)
    eigs = np.sort(eigvals(A))
    l0 = eigs[1]
    s = np.linalg.svd(A, compute_uv=False)
    z = np.zeros(K, dtype=complex)

    z1old = l0 + (epsilon / 2) * d0
    U, S, V = svd(z1old * np.eye(n, dtype=int) - A, full_matrices=True)
print(V); print("\n")
    umin = U[:, -1]
    smin = S[-1]
    vmin = V[-1, :]
    print(vmin); print()
    z1new = z1old - ((smin - epsilon) / np.real(np.conj(d0) * np.matmul(vmin.T, umin))) * d0
printf("\n z1new = ")
    print(z1new)
    count = 1
    while abs(z1new - z1old) > tol:
        z1old = z1new
        U, S, V = svd(z1old * np.eye(n, dtype=int) - A, full_matrices=True)
        umin = U[:, -1]
        smin = S[-1]
        vmin = V[-1, :]
        z1new = z1old - ((smin - epsilon) / np.real(np.conj(d0) * np.matmul(vmin.T, umin))) * d0
printf("\n z1new = ")
        print(z1new)
        count += 1
```

```
z[0] = z1new

    for k in range(1, K):
        U, S, V = svd(z[k - 1] * np.eye(n, dtype=int) - A, full_matrices=True)  # the triplet for z(k-1)
        umin = U[:, -1]
        smin = S[-1]
        vmin = V[-1, :]
        grad = np.matmul(vmin.T, umin)
        r = (grad * 1j) / np.abs(grad)
        pred = z[k - 1] + t * r
        # Correction
        ## z[k] = pred - ((smin - epsilon) / (np.matmul(umin.T, vmin)))
        z[k] = pred
        if k%100000 == 0:
            print(grad)
            print(r)
            print(pred)
            print(z[k])
            print("\n\n")
printf("\n\n\n iteration = %d \n\n\n\n", k)

    printf("\n")
plt.plot(z)

    if save == 1:
plt.plot(eigs, linestyle='none', marker='*')
plt.savefig(name, format='jpg')
```

```
## program

A = np.array([[0.0760, 0.4173, 0.4893, 0.7803, 0.1320],
              [0.2399, 0.0497, 0.3377, 0.3897, 0.9421],
              [0.1233, 0.9027, 0.9001, 0.2417, 0.9561],
              [0.1839, 0.9448, 0.3692, 0.4039, 0.5752],
              [0.2400, 0.4909, 0.1112, 0.0965, 0.0598]])

epsilons = [0.8, 0.5, 0.2, 0.1]

K = 50000
d0 = complex(1,1)
tol = 10**-6
t = 10**-3

for epsilon in epsilons:
    if epsilon == 0.8:
printf("\n calling bruhl with epsilon = %.2f \n", epsilon);
bruhl(A, epsilon, K, d0, tol, t, 1, 'random.jpg')
```

Τα αποτελέσματα ήταν παρόμοια με αυτά του Matlab.

# 7. Βιβλιογραφία

[ΔΕΣΥΠΡΗ2018]Δεσύπρη Αλεξάνδρα, Αλγοριθμικός Σχεδιασμός Ψευδοφάσματος Τετραγωνικών Πινάκων, Διπλωματική Εργασία, ΕΜΠ, 2018

[BEKAS2001] C. Bekas and E. Gallopoulos. Cobra: Parallel path following for computing thematrix pseudospectrum. Parallel Computing, 27(14):1879-1896, 2001.

[BEKAS2002] C. Bekas and E. Gallopoulos. Parallel computation of pseudospectra by fast descent.Parallel Computing, 28(2):223-242, 2002.

[BRUHL1996] M. Brühl. A curve tracing algorithm for computing the pseudospectrum. BIT,33(3):441-445, 1996.

[MEZHER2002a] D. Mezher and B. Philippe. Parallel computation of the pseudospectrum of largematrices. Parallel Computing, 28(2): 199-221, 2002.

[MEZHER2002b] D. Mezher and B. Philippe. PAT - a reliable path following algorithm. NumericalAlgorithms, 1(29):131-152, 2002.

[TREFETHEN2005] Trefethen, Embree, Spectra and Pseudospectra, Princeton University Press (2005)