ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΠΑΛΟΙΦΗ ΑΓΝΩΣΤΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΗΣ DIXON RESULTANT

Διπλωματική Εργασία

Τσιαγκάλης Χαράλαμπος

Επιβλέπουσα Καθηγήτρια:
Δασκαλοπούλου Ασπασία

Βόλος 2020

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΠΑΛΟΙΦΗ ΑΓΝΩΣΤΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΗΣ DIXON RESULTANT

Διπλωματική Εργασία

Τσιαγκάλης Χαράλαμπος

Επιβλέπουσα Καθηγήτρια:
Δασκαλοπούλου Ασπασία

Βόλος 2020

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ELIMINATION OF UNKNOWNS USING THE DIXON RESULTANT

Diploma Thesis

Tsiagkalis Charalampos

Supervisor:
Daskalopoulou Aspassia

Volos 2019

ΕΥΧΑΡΙΣΤΙΕΣ

Με αφορμή την παρούσα διπλωματική εργασία θα ήθελα να ευχαριστήσω ιδιαιτέρως την επιβλέπουσα καθηγήτρια κ. Δασκαλοπούλου Ασπασία για την καθοδήγηση και τις χρήσιμες συμβουλές που προσέφερε, όπως επίσης και τον ομότιμο καθηγητή κ. Ακρίτα Αλκιβιάδη, χωρίς τη βοήθεια του οποίου η εργασία αυτή δεν θα μπορούσε να πραγματοποιηθεί.

Επίσης, θα ήθελα να ευχαριστήσω ττην οικογένεια και τους φίλους μου για την ψυχολογική και οικονομική υποστήριξη κατά τη διάρκεια των σπουδών μου.

ΠΕΡΙΛΗΨΗ

Στόχος της παρούσας διπλωματικής εργασίας είναι η ανάδειξη της σημασίας της μεθόδου Dixon resultant [**Dixon1909**] για την απαλοιφή αγνώστων σε ένα σύστημα πολυωνυμικών εξισώσεων, καθώς και της τροποποίησής της από τους Kapur, Saxena και Yang[**Kapur1994**]. Επιπλέον, μέρος της εργασίας αποτελεί και η υλοποίηση του αλγορίθμου των τελευταίων σε python και η ενσωμάτωσή του στο εργαλείο υπολογιστικής άλγεβρας sympy. Το περιεχόμενο της εργασίας οργανώνεται ως εξής:

Στο Κεφάλαιο 1 παραθέτονται ορισμοί και εισαγωγικά στοιχεία για τις resultants και τη χρήση τους.

Στο Κεφάλαιο 2 παρουσιάζονται η κλασική Dixon Resultant, η γενίκευση της μεθόδου Cayley-Bézout από τον Dixon και μερικά παραδείγματα.

Τέλος, στο Κεφάλαιο 3 παρουσιάζεται η τροποποίηση της μεθόδου του Dixon από τους Kapur, Saxena και Yang και τα πλεονεκτήματά της.

ABSTRACT

The purpose of this thesis is to highlight the importance of the Dixon resultant for the elimination of unknowns in systems of polynomial equations, as well as of its variant developed by Kapur, Saxena and Yang. Part of this paper is also the implementation of the KSY algorithm in Python and its integration to SymPy, one of the most popular computer algebra systems (CAS). The material of this paper is organized as follows:

Chapter 1 lists definitions and a small introduction about resultants and their use.

Chapter 2 deals with the classical Dixon resultant, the generalization of the Cayley-Bézout method by Dixon and presents some examples.

Finally, Chapter 3 introduces the variation of Dixon's method by Kapur, Saxena and Yang and its advantage over the classical.

ΠΙΝΑΚΑΣ ΠΕΡΙΟΧΟΜΕΝΩΝ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Σήμερα, υπάρχει ουσιαστική ανάγκη για γρήγορη και αποδοτική λύση συστημάτων πολυωνυμικών εξισώσεων. Τέτοια συστήματα κάνουν την εμφάνισή τους σε διάφορους επιστημονικούς τομείς, όπως είναι η ρομποτική, η μηχανική, η κινηματική, η γεωδαισία και άλλοι. Μία κλασική μέθοδος για την λύση αυτών των συστημάτων είναι με τη χρήση των resultants. Η δυσκολία για τη λύση πολυωνύμων, πέρα από την εύρεση των ριζών τους, είναι η εύρεση των συνθηκών για τους συντελεστές (ή παραμέτρους) όπου το σύστημα συναντάται με ένα σύνολο λύσεων. Αυτές οι συνθήκες ονομάζονται resultant του συστήματος. Γενικά, μία resultant είναι ένα πολυώνυμο που προκύπτει από ένα σύστημα πολυωνυμικών εξισώσεων και περιλαμβάνει τη λύση του συστήματος [**Lewis1996**]. Με τον όρο λύση εννοούνται τα κοινά μηδενικά των πολυωνύμων που ανήκουν στο σύστημα. Μία θεωρία για την εύρεση της resultant και τη λύση του συστήματος είναι η απαλοιφή των μεταβλητών. Υπάρχουν δύο κύριες παραλλαγές για την απαλοιφή μεταβλητών από ένα πολυωνυμικό σύστημα σε μορφή πινάκων και τον υπολογισμό της resultant. Η Sylvester resultant [**Zhao2010**] -με την οποία δεν θα ασχοληθούμε- και η Bézout-Cayley resultant [**Palncz2013**, **Bezout1779**]. Και οι δύο μέθοδοι έχουν ως στόο την απαλοιφή ν μεταβλητών από $\nu + 1$ πολυώνυμα μέσω του πίνακα της resultant. Στο κεφάλαιο που ακολουθεί παρουσιάζεται η Dixon resultant που είναι τύπου Bézout-Cayley.

ΚΕΦΑΛΑΙΟ 2

Η κλασική Dixon Resultant

2.1 Η διατύπωση του Cayley για τη μέθοδο του Bézout

Αρχικά, ας υπενθυμίσουμε τη διατύπωση του Cayley [**Cayley1865**] για τη μέθοδο του Bézout για τη λύση συστημάτων δύο πολυωνυμικών εξισώσεων. Αξίζει να επισημάνουμε ότι όπως αναφέρει ο καθηγητής Kapur, η μέθοδος αυτή ανήκει στον Euler.

Έστω $f(x)$ και $g(x)$ πολυώνυμα, με βαθμό $deg = max(degree(f), degree(g))$, και α βοηθητική μεταβλητή. Η ποσότητα:

$$\delta(x,\alpha) = \frac{1}{(x-\alpha)} \begin{vmatrix} f(x) & g(x) \\ f(\alpha) & g(\alpha) \end{vmatrix}$$

είναι ένα συμμετρικό πολυώνυμο των x και α βαθμού $deg - 1$, το οποίο ονομάζουμε πολυώνυμο Dixon των f και g. Ο Cayley (και ο Bézout σε διαφορετική διατύπωση) παρατήρησαν ότι κάθε κοινό μηδενικό των f και g είναι και μηδενικό της $\delta(x,\alpha)$ για κάθε τιμή της α. Ως εκ τούτου, σε κάθε κοινό μηδενικό, κάθε συντελεστής της α^i στη $\delta(x,\alpha)$ είναι ≡ 0. Σε μορφή πινκάκων:

$$M \begin{bmatrix} 1 \\ x \\ \vdots \\ V_{deg} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

όπου, οι στήλες του $deg \times deg$ πίνακα M αποτελούνται από τους συντελεστές των α^i. Αυτό μας δίνει ένα ομογενές σύστημα με νέες μεταβλητές $v_1, v_2, \ldots v_{deg}$ που αντιστοιχούν στις $x^0, x^1, \ldots, x^{deg-1}$ και εξισώσεις που αντιστοιχούν στους συντελεστές

των α^i.

$$M \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ V_{deg} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.1}$$

Το παραπάνω σύστημα έχει μη τετριμμένες λύσεις αν και μόνο αν η ορίζουσά του D είναι μηδέν. Η D ονομάζεται Dixon resultant των f και g και M είναι ο πίνακας Dixon. Καταλήγουμε λοιπόν ότι ο μηδενισμός της Dixon resultant D είναι αναγκαία συνθήκη για την ύπαρξη κοινού μηδενικού των f και g. Η μορφή του Cayley για τη μέθοδο του Bézout παρουσιάζεται στα Παραδείγματα 1-4 που ακολουθούν. Υπολογίζουμε τα M και D για τα δεδομένα f και g και σχολιάζουμε τη σχέση μεταξύ του μηδενισμού της D και των κοινών μηδενικών του συστήματος $f = 0, g = 0$. Σε κάθε περίπτωση παραγοντοποιούμε τα f και g για εύκολο έλεγχο.

2.2 Παράδειγμα 1: Αριθμητικοί συντελεστές – Κοινή ρίζα

Έστω πολυώνυμα f και g :

$$f(x) = x^3 - 2x^2 - 11x + 12 = (x-4)(x-1)(x+3)$$
$$g(x) = x^2 + 3x - 4 = (x-1)(x+4)$$

Το πολυώνυμο Dixon θα είναι:

$$\delta(x,\alpha) = \frac{1}{x-\alpha} \begin{vmatrix} x^3 - 2x^2 - 11x + 12 & x^2 + 3x - 4 \\ \alpha^3 - 2\alpha^2 - 11\alpha + 12 & \alpha^2 + 3\alpha - 4 \end{vmatrix} =$$

$$(x^2 + 3x - 4)\alpha^2 + (3x^2 + x - 4)\alpha^1 + (-4x^2 - 4x + 8)\alpha^0$$

Εξισώνοντας τους συντελεστές των απαίρνουμε τις παρακάτω εξισώσεις:

$$x^2 + 3x + -4 = 0$$
$$3x^2 + x + -4 = 0$$
$$4x^2 - 4x - 8 = 0$$

Οπότε ο πίνακας Dixon M θα είναι:

$M = \begin{bmatrix} 1 & 3 & -4 \\ 3 & 1 & -4 \\ -4 & -4 & 8 \end{bmatrix}$ και η Dixon Reusltant: $D = \begin{vmatrix} 1 & 3 & -4 \\ 3 & 1 & -4 \\ -4 & -4 & 8 \end{vmatrix} = 0$

Εδώ παρατηρούμε ότι το σύστημα έχει μια κοινή ρίζα στο $x = 1$, γι' αυτό και η ορίζουσα βγαίνει 0. Καθώς η D εξισώνεται ταυτοτικά με το 0, δεν μπορούμε να πάρουμε κάποια επιπλέον πληροφορία για την εύρεση της ρίζας.

2.3 Παράδειγμα 2: Αριθμητικοί συντελεστές – Καμία κοινή ρίζα

Έστω πολυώνυμα f και g :

$$f(x) = x^3 - 2x^2 - 11x + 12 = (x-4)(x-1)(x+3)$$
$$g(x) = x^2 + 5x + 4 = (x+1)(x+4)$$

Τότε, το πολυώνυμο Dixon θα είναι:

$$\left(x^2 + 5x + 4\right)\alpha^2 + \left(5x^2 + 5x - 20\right)\alpha^1 + \left(4x^2 - 20x - 104\right)\alpha^0$$

Ο πίνακας Dixon και η ορίζουσά του:

$$M = \begin{bmatrix} 1 & 5 & 4 \\ 5 & 5 & -20 \\ 4 & -20 & -104 \end{bmatrix} \text{ και } D = \begin{vmatrix} 1 & 5 & 4 \\ 5 & 5 & -20 \\ 4 & -20 & -104 \end{vmatrix} = 800$$

Η ορίζουσα είναι μη μηδενική, πράγμα που σημαίνει ότι οι δύο εξισώσεις δεν έχουν κοινή ρίζα.

2.4 Παράδειγμα 3: Παραμετρικοί συντελεστές - Εντοπισμός όλων των ριζών

Έστω πολυώνυμα f και g :

$$f(x) = -A^3 - 2A^2 + Ax^2 - Ax + 3A + 3x^2 - 3x = (-A + x)(A + 3)(A + x - 1)$$
$$g(x) = 3A^2 + 4Ax + x^2 = (A + x)(3A + x)$$

Το πολυώνυμο Dixon θα είναι:

$$(4A^3 + 4A^2x + 11A^2 + 13Ax - 3A + 3x)\,\alpha^1 + (4A^4 + 4A^3x + 5A^3 + 11A^2x - 21A^2 - 3Ax)\,\alpha^0$$

Ο πίνακας Dixon θα είναι:

$$M = \begin{bmatrix} 4A^2 + 13A + 3 & 4A^3 + 11A^2 - 3A \\ 4A^3 + 11A^2 - 3A & 4A^4 + 5A^3 - 21A^2 \end{bmatrix}$$ και

$$D = \begin{vmatrix} 4A^2 + 13A + 3 & 4A^3 + 11A^2 - 3A \\ 4A^3 + 11A^2 - 3A & 4A^4 + 5A^3 - 21A^2 \end{vmatrix} = -8A^2(A + 3)^2(2A + 1)$$

Θέτοντας $D = 0$ παίρνουμε τις εξής λύσεις: $A = -3, -3, -\frac{1}{2}, 0, 0$

Στην πραγματικότητα οι λύσεις των $f(x, A) = 0$ και $g(x, A) = 0$ στο πεδίο των ρητών είναι:

$$(x, A) = (9, -3), (3, -3), (0, 0), (\tfrac{3}{2}, -\tfrac{1}{2})$$

Παρατηρούμε ότι οι τιμές της παραμέτρου A από τη λύση του συστήματος των δύο παραπάνω εξισώσεων είναι και οι ρίζες της D. Ωστόσο, η λύση $A = 0$ έχει πολαπλότητα 2 στην εξίσωση $D = 0$, ενώ πολαπότητα 1 στο σύστήμα των δύο εξισώσεων. Σε αυτό το παράδειγμα, όλα τα μηδενικά του συστήματος εντοπίστηκαν από τα μηδενικά της Dixon resultant.

2.5 Παράδειγμα 4: Παραμετρικοί συντελεστές – Χωρίς ανάκτηση ριζών

Έστω πολυώνυμα f και g :

$$f(x) = -A^2x + A^2 + Ax^2 - 4Ax + 3A + 3x^2 - 3x = (-A + x)(A + 3)(x - 1)$$
$$g(x) = Ax - A + x^2 - x = (A + x)(x - 1)$$

Το πολυώνυμο Dixon:

$$\left(2A^2x - 2A^2 + 6Ax - 6A\right)\alpha^1 - \left(2A^2x + 2A^2 - 6Ax + 6A\right)\alpha^0$$

Ο πίνακας Dixon και η ορίζουσά του:

$M = \begin{bmatrix} 2A^2 + 6A & -2A^2 - 6A \\ -2A^2 - 6A & 2A^2 + 6A \end{bmatrix}$ και $D = \begin{vmatrix} 2A^2 + 6A & -2A^2 - 6A \\ -2A^2 - 6A & 2A^2 + 6A \end{vmatrix} = 0$

Στην πραγματικότητα οι λύσεις των $f(x, A) = 0$ και $g(x, A) = 0$ στο πεδίο των ρητών είναι:

$(x, A) = (3, -3), (1, A), (0, 0)$

Αυτή τη φορά, ο μηδενισμός της D δεν δίνει κάποια επιπλέον πληροφορία για τις κοινές ρίζες των εξισώσεων.

2.6 Η γενίκευση της μεθόδου Cayley-Bézout από τον Dixon

Ο Dixon γενίκευσε την προσέγγιση του Cayley για τη μέθοδο του Bézout για συστήματα τριών πολυωνυμικών εξισώσεων με δύο αγνώστους.

Έστω τρία πολυώνυμα f, g, h με αγνώστους τις μεταβλητές x, y:

$$f(x,y) = 0$$
$$g(x,y) = 0$$
$$h(x,y) = 0$$

Το πολυώνυμο Dixon τώρα ορίζεται ως εξής:

$$\delta(x,y,\alpha,\beta) = \frac{1}{(x-\alpha)(y-\beta)} \begin{vmatrix} f(x,y) & g(x,y) & h(x,y) \\ f(\alpha,y) & g(\alpha,y) & h(\alpha,y) \\ f(\alpha,\beta) & g(\alpha,\beta) & h(\alpha,\beta) \end{vmatrix}$$

όπου α, β βοηθητικές μεταβλητές.

Θέτοντας τα γινόμενα $\alpha^i\beta^j$ ίσα με μηδέν παίρνουμε ένα ομογενές γραμμικό σύστημα όπως και πριν. Η αντίστοιχη ορίζουσα του πίνακα των συντελεστών είναι η Dixon resultant D.

Ο Dixon απέδειξε ότι για την ύπαρξη κοινών ριζών σε τρία γενικευμένα πολυώνυμα δευτέρου βαθμού, ο μηδενισμός της D είναι αναγκαία συνθήκη. Επιπλέον, η D δεν είναι ταυτοτικά μηδέν.

Η μέθοδος και οι αποδείξεις του Dixon εύκολα γενικεύονται για ένα σύστημα $\nu + 1$ πολυωνύμων ν βαθμού με ν αγνώστους. Ας θυμηθούμε ότι ένα πολυώνυμο είναι γενικευμένο αν όλοι οι συντελεστές του είναι ανεξάρτητες μεταξύ τους παράμετροι. Ένα πολυώνυμο ν μεταβλητών είναι ν-οστού βαθμού αν εμφανίζονται σε αυτό όλες οι δυνάμεις των μεταβλητών του από τη μηδενική έως και τη μέγιστη.

Έστω $p_j(x_1, \ldots, x_\nu), j = 1, \ldots, \nu + 1$ είναι $\nu + 1$ πολυώνυμα ν μεταβλητών.

$$\sum_{i_1=0_1}^{k} \cdots \sum_{i_\nu=0_\nu}^{k} \alpha_{j,i_1,\ldots,i_\nu} x_{1_1}^{i} \ldots x_{\nu_\nu}^{i}, 1 \leq j \leq \nu + 1$$

όπου α είναι διακριτοί απροσδιόριστοι συντελεστές.

Για παράδειγμα, τα τρία παρακάτω πολυώνυμα είναι γενικευμένα ν-στου βαθμού $(x : 2, y : 1)$:

$$p_1 = a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 x^2 y$$
$$p_2 = b_0 + b_1 x + b_2 y + b_3 xy + b_4 x^2 + b_5 x^2 y$$
$$p_3 = c_0 + c_1 x + c_2 y + c_3 xy + c_4 x^2 + c_5 x^2 y$$

Αντίθετα, τα τρία παρακάτω πολυώνυμα δεν είναι γενικευμένα:

$$q_1 = 3x + (\alpha x)^2 + x^2 y$$
$$q_2 = y + xy - (\alpha x)^2$$
$$q_3 = b + c_1 x + c_2 y$$

Στο q_1 ο συντελεστής 3 δεν είναι απροσδιόριστος -οπότε, δεν είναι γενικευμένο- και ο συντελεστής του $x^0 y^0$ λείπει, οπότε δεν είναι πλήρες ν-στού βαθμού.

Η μέθοδος του Dixon εφαρμόζεται στα πολυώνυμα με συμβολικές μεταβλητές, οι οποίες επιτρέπουν την ταυτόχρονη απαλοιφή μιας ομάδας αγνώστων με έναν μόνο υπολογισμό. Το γεγονός αυτό, μαζί με το σχετικά μικρό μέγεθος των τελικών ορίζουσών (συγκριτικά με άλλες μεθόδους που χρησιμοποιούν resultants) κάνουν τη μέθοδο αυτή πολύ αποδοτική. Δυστυχώς, αν τα πολυώνυμα δεν είναι γενικευμένα και ν-στού βαθμού, δεν θα έχουμε σωστές λύσεις. Στα παραδείγματα που ακολουθούν, παρουσιάζονται η μέθοδος του Dixon και οι περιορισμοί της για πολυώνυμα που δεν είναι απαραίτητα γενικευμένα και ν-στού βαθμού.

2.7 Παράδειγμα 5: Μη μηδενική ορίζουσα

Έστω τα πολυώνυμα:

$$f(x,y) = xy + 3x - 3y - 9 = (x-3)(y+3)$$
$$g(x,y) = xy - 3x + 3y - 9 = (x+3)(y-3)$$
$$h(x,y) = xy + 2x - 2y - 4 = (x-2)(y+2)$$

Το πολυώνυμο Dixon είναι $30\alpha^1 + 30y\beta^1$ και ο πίνακας Dixon:

$$M = \begin{bmatrix} 0 & 30 \\ 30 & 0 \end{bmatrix}$$

Η Dixon Resulatant είναη μη μηδενική $D = det(M) = -900$. Επομένως δεν περιμένουμε να βρούμε κοινές ρίζες.

2.8 Παράδειγμα 6: Μηδενική ορίζουσα - Χωρίς ανάκτηση ριζών

Έστω το σύστημα:

$$f(x,y) = xy + 3x - y^2 - 3y = (x-y)(y+3)$$
$$g(x,y) = x^2 + 4xy + 3y^2 = (x+y)(x+3y)$$
$$h(x,y) = x^2 - xy - 2y^2 = (x-2y)(x+y)$$

Το πολυώνυμο Dixon είναι:
$(-15x + 10y^2 + 15y)\ \alpha^2\beta^0 + (-10xy - 15x + 15y)\ \alpha^1\beta^1 + (-15xy - 15y^2)\alpha^1\beta^0 + (15xy + 15y^2)\ \alpha^0\beta^1$

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -15 & 10 & 15 & 0 \\ 0 & -10 & -15 & 0 & 15 & 0 \\ 0 & -15 & 0 & -15 & 0 & 0 \\ 0 & 15 & 0 & 15 & 0 & 0 \end{bmatrix}$$

Η ορίζουσα του πίνακα μας δίνει $D = 0$, οπότε δεν μπορούμε να πάρουμε κάποια πληροφορία για τις ρίζες του συστήματος, οι οποίες στην πραγματικότητα είναι οι $(x, y) = (0, 0), (3, -3)$

Παρατηρούμε ότι αν αφαιρέσουμε τις μηδενικές γραμμές και στήλες του πίνακα, όπως θα δούμε και στη μέθοδο του επόμενου κεφαλαίου, μειώνουμε το μέγεθος του πίνακα χωρίς όμως να επηρεάσουμε την τιμή της ορίζουσας. Υπάρχει όμως πιθανότητα ο πίνακας να μην είναι πλέον τετραγωνικός και η ορίζουσα να μην ορίζεται.

2.9 Παράδειγμα 7: Μη μηδενική ορίζουσα - Χωρίς ανάκτηση ριζών

$$f(x, y) = xy - 3y = y\,(x - 3)$$
$$g(x, y) = xy - 3x = x\,(y - 3)$$
$$h(x, y) = xy - 2y = y\,(x - 2)$$

Το πολυώνυμο Dixon είναι $3y\alpha^1\beta^0$

Ο πίνακας Dixon είναι $M = \begin{bmatrix} 0 & 0 \\ 0 & 3 \end{bmatrix}$ και αν αφαιρέσουμε τις μηδενικές γραμμές και στήλες του παίρνουμε τον $M = \begin{bmatrix} 3 \end{bmatrix}$

Η ορίζουσα του νέου πίνακα είναι μη μηδενική, χωρίς αυτό να σημαίνει ότι το σύστημα δεν έχει λύσεις. Στην πραγματικότητα έχει την τετριμμένη.

Στο παράδειγμα αυτό είδαμε ότι ο μηδενισμός της ορίζουσας του νέου πίνακα (αυτού δηλαδή που έχει δημιουργηθεί αφαιρόντας τις μηδενικές γραμμές και στήλες από τον αρχικό πίνακα Dixon) δεν είναι αναγκαία συνθήκη για την ύπαρξη κοινών λύσεων στο σύστημα. Αυτό, συμβαίνει γιατί η ορίζουσα του αρχικού πίνακα Dixon είναι 0 όπως περιμέναμε.

2.10 Παράδειγμα 8: Ορίζουσα ταυτοτικά μηδέν - Χωρίς ανάκτηση ριζών

Έστω τα πολυώνυμα f, g, h των μεταβλητών x, y:

$$f(x,y) = -A^2 + Ax - A + xy + x + y^2 + y = (-A + x + y)(A + y + 1)$$
$$g(x,y) = Ax + Ay + x^2 + xy - x - y = (x + y)(A + x - 1)$$
$$h(x,y) = -Ax - Ay - 2A + x^2 + xy + 2x = (-A + x)(x + y + 2)$$

Το πολυώνυμο Dixon είναι:

$$(2Ax + 2Ay + 2A - 3x - y)\,\alpha^2\beta^0 +$$
$$(2Ax + 2Ay + 2A - 3x - y)\,\alpha^1\beta^1 +$$
$$(-2A^2x - 2A^2y + 2A^2 + 2Axy + 3Ax + 2Ay^2 + 5Ay - 3xy - 2x - 3y^2 - 4y)\,\alpha^1\beta^0 +$$
$$(2A^2 + 2Axy + 2Ax + 2Ay^2 + 4Ay - 2A - xy - y^2 - 2y)\,\alpha^0\beta^1 +$$
$$(2A^3x + 2A^3y + A^2x + A^2y + 2A^2 + 2Axy + Ax + 2Ay^2 + Ay - 2A)\,\alpha^0\beta^0$$

Ο πίνακας Dixon που προκύπτει είναι:

$$M = \begin{bmatrix} 0 & 2A-3 & 0 & 2A-1 & 2A \\ 0 & 2A-3 & 0 & 2A-1 & 2A \\ 2A-3 & -2A^2+3A-2 & 2A-3 & -2A^2+5A-4 & 2A^2 \\ 2A-1 & 2A & 2A-1 & 4A-2 & 2A^2-2A \\ 2A & 2A^3+A^2+A & 2A & 2A^3+A^2+A & 2A^2-2A \end{bmatrix}$$

Ο υπολογισμό της ορίζουσάς του μας δίνει $D = 0$

Στην πραγματικότητα όμως το σύστημα $f(x,y,A) = 0$, $h(x,y,A) = 0$, $g(x,y,A) = 0$ έχει πολλές λύσεις στο πεδίο Q, και συγκεκριμένα:

$$(x,y,A) = (0,-2,-1), (3,-5,-2), (0,0,0), (\tfrac{1}{2}, -\tfrac{3}{2}, \tfrac{1}{2}), (\tfrac{1}{2}, 0, \tfrac{1}{2})$$

Βλέπουμε λοιπόν ότι η resultant είναι 0 όπως περιμέναμε, όμως επειδή είναι ταυτοτικά ίση με το 0 δεν μπορούμε να πάρουμε κάποια πληροφορία για τις ρίζες του

συστήματος.

2.11 Παράδειγμα 9: Μη τετραγωνικός πίνακας

Έστω τα πολυώνυμα f, g, h των μεταβλητών x, y :

$$
\begin{aligned}
f(x,y) &= -74x^2 - 83xy^4 - 332xy^3 - 593xy^2 - 522xy - 30x + 126y^4 + 504y^3 + \\
&\quad + 910y^2 + 909y + 303 \\
g(x,y) &= 77x^4y + 77x^4 - 308x^3y - 308x^3 + 462x^2y + 462x^2 + 47xy^2 - 295xy - \\
&\quad - 403x - 2y^4 - 8y^3 - 59y^2 - 12y + 102 \\
h(x,y) &= 75x^2y^3 + 225x^2y^2 + 160x^2y + 10x^2 - 150xy^3 - 450xy^2 - 320xy - \\
&\quad - 111x + 72y^4 + 422y^3 + 834y^2 + 586y + 193
\end{aligned}
$$

Το πολυώνυμο Dixon είναι:

$(-427350x^2y - 427350x^2 - 479325xy^5 - 2396625xy^4 - 5341875xy^3 - 6439125xy^2 - 3187800xy - 173250x + 727650y^5 + 3638250y^4 + 8165850y^3 + 10504725y^2 + 6999300y + 1749825) \alpha^5\beta^2 + (-427350x^2y^2 - 1709400x^2y - 1282050x^2 - 479325xy^6 - 3834600xy^5 - 12531750xy^4 - 22464750xy^3 - 22505175xy^2 - 9736650xy - 519750x + 727650y^6 + 5821200y^5 + 19080600y^4 + 35002275y^3 + 38513475y^2 + 22747725y + 5249475) \alpha^5\beta + (-427350x^2y^3 - 1709400x^2y^2 - 2193730x^2y - 911680x^2 - 479325xy^6 - 3355275xy^5 - 10135125xy^4 - 16268175xy^3 - 13501950xy^2 - 4725490xy + 285978x + 727650y^6 + 5093550y^5 + 15032094y^4 + 24431869y^3 + 23256618y^2 + 12409397y + 2399936) \alpha^5 + (479325x^2y^4 + 1917300x^2y^3 + 2460535x^2y^2 + 1086470x^2y + 63910x^2 - 958650xy^4 - 3834600xy^3 - 4921070xy^2 - 2754521xy - 709401x + 460152y^5 + 3157154y^4 + 8027096y^3 + 9075220y^2 + 4978589y + 1233463) \alpha^4\beta^3 + (-427350x^3y - 427350x^3 + 1437975x^2y^4 + 4787860x^2y^3 + 4489485x^2y^2 + 3786090x^2y + 2646490x^2 + 2644950xy^5 + 10348800xy^4 + 19957630xy^3 + 26700674xy^2 + 14398615xy - 48279x + 460152y^6 + 631862y^5 - 1173788y^4 - 7811496y^3 - 21748881y^2 - 20847981y - 5565098) \alpha^4\beta^2 + (-427350x^3y^2 - 1709400x^3y - 1282050x^3 + 1022560x^2y^4 + 2162160x^2y^3 + 2048200x^2y^2 + 8537760x^2y + 7629160x^2 + 2644950xy^6 + 21159600xy^5 + 67162480xy^4 + 119955374xy^3 + 126658070xy^2 + 58650130xy + 3299604x + 460152y^7 + 631862y^6 - 10983896y^5 - 50743462y^4 - 111383965y^3 - 145094411y^2 - 95982579y - 22684277) \alpha^4\beta + (-427350x^3y^3 - 1709400x^3y^2 - 2193730x^3y - 911680x^3 + 63910x^2y^4 + 2710400x^2y^3 +$

$10275650x^2y^2+13131580x^2y+5502420x^2+2644950xy^6+18514650xy^5+54863193xy^4+$
$86304064xy^3+70744443xy^2+23560152xy-2466310x+460152y^7+171710y^6-11155606y^5-$
$43068102y^4-77586509y^3-79381148y^2-43752478y-7164311\ )\ \alpha^4+(\ 479325x^3y^4+$
$1917300x^3y^3+2460535x^3y^2+1086470x^3y+63910x^3-3603600x^2y^4-14414400x^2y^3-$
$18498480x^2y^2-9159997x^2y-1472317x^2+6146294xy^4+24058496xy^3+30048480xy^2+$
$17117947xy+4981669x-1840608y^5-13928684y^4-36454880y^3-39993184y^2-20752886y-$
$5126506\ )\ \alpha^3\beta^3+(\ 1437975x^3y^4+4787860x^3y^3+4489485x^3y^2+2076690x^3y+937090x^3-$
$10810800x^2y^4-35938980x^2y^3-34014442x^2y^2-16168537x^2y-7282275x^2-5786550xy^5-$
$11020548xy^4-7043036xy^3-27107953xy^2-20504329xy+4456036x-1840608y^6-$
$9803948y^5-34744876y^4-56945552y^3-15218480y^2+22290966y+7683980\ )\ \alpha^3\beta^2+($
$1022560x^3y^4+2162160x^3y^3+338800x^3y^2+1700160x^3y+2500960x^3-7687680x^2y^4-$
$16573942x^2y^3-1203972x^2y^2-6553778x^2y-14236068x^2-5786550xy^6-46292400xy^5-$
$139376622xy^4-248875037xy^3-291424749xy^2-151289397xy-10546013x-1840608y^7-$
$9803948y^6-14287516y^5+11444272y^4+101009488y^3+220465728y^2+182422790y+$
$45875202\ )\ \alpha^3\beta+(\ 63910x^3y^4+1001000x^3y^3+3438050x^3y^2+4356660x^3y+1855700x^3-$
$1472317x^2y^4-8754592x^2y^3-21518343x^2y^2-25687200x^2y-11451132x^2-5786550xy^6-$
$40505850xy^5-115696607xy^4-176478071xy^3-145618269xy^2-46316520xy+8941025x-$
$1840608y^7-7974440y^6-6390776y^5+21274484y^4+67046760y^3+92916434y^2+58570023y+$
$6028881\ )\ \alpha^3+(\ -1686300x^3y^4-6745200x^3y^3-8656340x^3y^2-4814117x^3y-1216677x^3+$
$9056894x^2y^4+35700896x^2y^3+44989560x^2y^2+25356331x^2y+7010773x^2-292575xy^5-$
$15340094xy^4-54840840xy^3-63270909xy^2-36168418xy-12690830x+12450y^7+$
$87150y^6+3304147y^5+25427621y^4+65927870y^3+67524016y^2+31881054y+8086734$
$\ )\ \alpha^2\beta^3+(\ -5058900x^3y^4-16787540x^3y^3-16056502x^3y^2-7861777x^3y-3533915x^3+$
$26644002x^2y^4+86518894x^2y^3+79268651x^2y^2+38296489x^2y+18902730x^2-292575xy^6+$
$4154250xy^5-12889096xy^4-51488017xy^3-978832xy^2+17374758xy-14782856x+$
$37350y^7+3441472y^6+21723087y^5+83643019y^4+142490813y^3+58887150y^2-28481214y-$
$11217255\ )\ \alpha^2\beta^2+(\ -3597440x^3y^4-7925302x^3y^3+151228x^3y^2+246862x^3y-4232228x^3+$
$18345558x^2y^4+37739317x^2y^3-9594431x^2y^2-7894425x^2y+21093765x^2+6202275xy^6+$
$50279400xy^5+144141112xy^4+276203076xy^3+393003445xy^2+229023462xy+14506904x+$
$2787472y^7+18602862y^6+50416424y^5+72726536y^4-15873643y^3-238239839y^2-$
$247350483y-65457587\ )\ \alpha^2\beta+(\ -1216677x^3y^4-4750592x^3y^3-7766143x^3y^2-8260560x^3y-$

$4028332x^3+7010773x^2y^4+25913503x^2y^3+39996495x^2y^2+43914178x^2y+22820413x^2+$
$6863475xy^6+48044325xy^5+130168029xy^4+191706133xy^3+162264788xy^2+40567889xy-$
$24256661x+2762572y^7+14716490y^6+28554768y^5+23077724y^4-20752021y^3-82552032y^2-$
$63531123y-2929270\ )\ \alpha^2+(\ 2311694x^3y^4+8720096x^3y^3+10364200x^3y^2+6099863x^3y+$
$2144065x^3-10546844x^2y^4-39226880x^2y^3-45149104x^2y^2-25237982x^2y-8768914x^2+$
$722475xy^5+18446511xy^4+58403790xy^3+57936282xy^2+28473358xy+10878354x-$
$24900y^7+106572y^6-1624218y^5-21112128y^4-58948210y^3-57868244y^2-24580203y-$
$5965255\ )\ \alpha\beta^3+(\ 6408402x^3y^4+19368734x^3y^3+15042643x^3y^2+6849381x^3y+4767070x^3-$
$28680036x^2y^4-83963572x^2y^3-57009876x^2y^2-18438882x^2y-16712542x^2+722475xy^6+$
$2265975xy^5+36859854xy^4+78026534xy^3-8818450xy^2-46206714xy+3024512x+$
$206172y^7-775380y^6-13830492y^5-75700164y^4-141233150y^3-58798771y^2+31051637y+$
$13765886\ )\ \alpha\beta^2+(\ 3955798x^3y^4+6038109x^3y^3-8989519x^3y^2-6906977x^3y+4164853x^3-$
$16469068x^2y^4-18195408x^2y^3+62229860x^2y^2+53231332x^2y-10724868x^2-2791350xy^6-$
$23987250xy^5-69381513xy^4-184856918xy^3-363795557xy^2-255517232xy-32605356x-$
$1816040y^7-14481014y^6-51990176y^5-108939782y^4-66614527y^3+150869517y^2+$
$202980662y+59323574\ )\ \alpha\beta+(\ 2144065x^3y^4+6911135x^3y^3+8931923x^3y^2+10871938x^3y+$
$6707085x^3-8768914x^2y^4-25481456x^2y^3-27437410x^2y^2-37389891x^2y-26665023x^2-$
$4447800xy^6-31134600xy^5-82469259xy^4-132413763xy^3-141013716xy^2-45903474xy+$
$20507352x-2426754y^7-15261148y^6-41533723y^5-67100691y^4-45657603y^3+33497021y^2+$
$44511071y+1729395\ )\ \alpha+(\ -1300068x^3y^4-4346496x^3y^3-3692304x^3y^2-838530x^3y-$
$192654x^3+5200272x^2y^4+17385984x^2y^3+14769216x^2y^2+3354120x^2y+770616x^2-$
$421050xy^5-9040608xy^4-25879184xy^3-19323874xy^2-2852048xy-449324x+22244y^7-$
$125164y^6-623520y^5+4955474y^4+19131760y^3+17437558y^2+3822228y-253632\ )\ \beta^3+($
$-3046428x^3y^4-6488636x^3y^3+3160696x^3y^2+8958642x^3y+2355738x^3+12185712x^2y^4+$
$25954544x^2y^3-12642784x^2y^2-35834568x^2y-9422952x^2-421050xy^6-2924400xy^5-$
$23059408xy^4-35382026xy^3+31878064xy^2+63055988xy+17972260x-214140y^7-$
$1501654y^6-2879042y^5+13302170y^4+35925312y^3-2565268y^2-38943672y-16967094$
$)\ \beta^2+(\ -645876x^3y^4+5957028x^3y^3+26271784x^3y^2+25603424x^3y+5934544x^3+$
$2583504x^2y^4-23828112x^2y^3-105087136x^2y^2-102413696x^2y-23738176x^2+22950xy^6+$
$1170000xy^5+6717879xy^4+68141590xy^3+198675425xy^2+177414890xy+43851904x+$
$25564y^7+666738y^6+5629718y^5+20805125y^4+2187508y^3-95110761y^2-119393743y-$

$40453733\ )\ \beta - 192654x^3y^4 + 2163084x^3y^3 + 8290282x^3y^2 + 6097861x^3y + 163317x^3 + 770616x^2y^4 - 8652336x^2y^3 - 33161128x^2y^2 - 24391444x^2y - 653268x^2 + 1009350xy^6 + 7065450xy^5 + 21411831xy^4 + 54431599xy^3 + 92970694xy^2 + 60248446xy + 6537954x + 641590y^7 + 4248464y^6 + 13144377y^5 + 24692606y^4 + 13936499y^3 - 31218293y^2 - 39502772y - 9687417$

Ο πίνακας Dixon που προκύπτει στην περίπτωση αυτή είναι $23x25$. Οπότε είναι μη-τετραγωνικός και η ορίζουσά του δεν ορίζεται.

2.12 Πιθανά προβλήματα με τη Μέθοδο του Dixon

Η μέθοδος του Dixon εφαρμόζεται μόνο σε γενικευμένα πολυώνυμα ν βαθμού. Αν δεν ικανοποιείται αυτή η συνθήκη, τότε μπορεί να έρθουμε αντιμέτωποι με τα εξής προβλήματα:

Α) Ο πίνακας Dixon μπορεί να είναι ιδιόμορφος, δηλαδή να έχει μηδενική ορίζουσα (Παραδείγματα 6 και 8).

Β) Μετά την αφαίρεση των μηδενικών γραμμών και στηλών ο μηδενισμός της ορίζουσας του πίνακα Dixon μπορεί να μην αποτελεί αναγκαία συνθήκη για την ύπαρξη κοινών μηδενικών (Παράδειγμα 7).

Γ) Μετά την αφαίρεση των μηδενικών γραμμών και στηλών, ο πίνακας Dixon μπορεί να μην είναι τετραγωνικός και η ορίζουσά του να μην ορίζεται (Παράδειγμα 9).

ΚΕΦΑΛΑΙΟ 3

Η προσέγγιση Kapur–Saxena–Yang

3.1 Εισαγωγή

Οι Kapur, Saxena και Yang αντιμετώπισαν επιτυχώς και τα τρία από τα παραπάνω προβλήματα, δεδομένου ότι ισχύει μία συνθήκη που θα εξηγήσουμε παρακάτω. Ας περιγράψουμε το κύριο θεώρημα και τον αλγόριθμό τους.

Έστω ότι έχουμε ένα σύστημα $\nu + 1$ πολυωνυμικών εξισώσεων με ν μεταβλητές, τέτοιο ώστε οι συντελεστές των πολυωνύμων να είναι και αυτοί πολυώνυμα με πεπερασμένο σύνολο παραμέτρων. Έστω M ο πίνακας Dixon που υπολογίζεται όπως προηγουμένως.

Έστω επίσης, M' μία κλιμακωτή μορφή του πίνακα που παράγεται από τον M χρησιμοποιώντας βασικές γραμμοπράξεις εκτός από πολλαπλασιασμό γραμμών. Μετατροπή σε τέτοια μορφή είναι πάντοτε δυνατή. Έστω D το γινόμενο οδηγών στοιχείων του M'.

Η Συνθήκη: Η στήλη που αντιστοιχεί στο μονώνυμο $1 = x_1^0 x_2^0 \dots x_\nu^0$ του πίνακα Dixon πρέπει να είναι γραμμικώς ανεξάρτητη από τις υπόλοιπες.

Η στήλη αυτή είναι η τελευταία στήλη του πίνακα Dixon.

Θεώρημα: Αν η συνθήκη είναι αληθής, τότε η εξίσωση $D = 0$ είναι αναγκαία συνθήκη για την ύπαρξη κοινών μηδενικών.

Από το θεώρημα αυτό προκύπτει ένας απλός αλγόριθμος για την ικανοποίηση της συνθήκης $D = 0$, την οποία ονομάζουμε Kapur-Saxena-Yang Dixon resultant.

3.2 Αλγόριθμος (Υπολογισμός της Kapur-Saxena-Yang Dixon resultant)

Είσοδος: Ένα σύνολο πολυωνύμων, με αριθμητικούς ή παραμετρικούς συντελεστές.

1) Υπολογίζουμε τον πίνακα Dixon M. Αν ισχύει η συνθήκη, τότε συνεχίζουμε.

2) Υπολογίζουμε την ανηγμένη κλιμακωτή μορφή $'$ του πίνακα M χωρίς να πολλαπλασιάσουμε τις γραμμές του.

3) Υπολογίζουμε το γινόμενο D των οδηγών στοιχείων του M'.

Έξοδος: Η D είναι η Kapur-Saxena-Yang resultant. Ο μηδενισμός της είναι αναγκαία συνθήκη για την ύπαρξη λύσεων για το δοθέν σύστημα.

3.3 Έλεγχος για την ορθότητα της Συνθήκης:

Έστω M ο πίνακας Dixon με στήλες $m_1, m_2, \ldots, m_\nu$. Έστω, επίσης, $w = \begin{bmatrix} w_1, \ldots, w_\nu \end{bmatrix}^T$ η λύση του συστήματος $Mw = 0 \Leftrightarrow w_1 m_1 + w_2 m_2 + \cdots + w_\nu m_\nu = 0$.

Τότε, η στήλη m_ν είναι γραμμικώς ανεξάρτητη από τις στήλες $m_1, m_2, \ldots, m_{\nu-1}$ αν και μόνο αν $w_\nu = 0$.

Βλέπουμε λοιπόν ότι μπορούμε εύκολα να ελέγξουμε την ορθότητα της συνθήκης στο δεύτερο βήμα του αλγορίθμου. Συγκεκριμένα, μπορούμε να πούμε ότι η Συνθήκη ισχύει αν η τελευταία γραμμή του πίνακα M' είναι της μορφής $\begin{bmatrix} 0 & 0 & \ldots & 1 \end{bmatrix}$.

Ας ξαναδούμε τα Παραδείγματα 3, 4, 8, 9 υπολογίζοντας τώρα την Kapur-Saxena-Yang Dixon resultant, D.

3.4 Παράδειγμα 10: Επέτκαση του παραδείγματος 3

Ο πίνακας Dixon που υπολογίσαμε στο Παράδειγμα 3 ήταν:

$$M = \begin{bmatrix} 4A^2 + 13A + 3 & 4A^3 + 11A^2 - 3A \\ 4A^3 + 11A^2 - 3A & 4A^4 + 5A^3 - 21A^2 \end{bmatrix}$$

Αφού η δεύτερη στήλη του πίνακα Dixon δεν είναι βαθμωτό πολλαπλάσιο της πρώτης, η Συνθήκη του Θεωρήματος ισχύει. Οπότε, μπορούμε να συνεχίσουμε παίρνοντας την ανηγμένη κλιμακωτή μορφή του M:

$$M' = \begin{bmatrix} 4A^2 + 13A + 3 & A\left(4A^2 + 11A - 3\right) \\ 0 & -\frac{A^2\left(16A^2+56A+24\right)}{4A+1} \end{bmatrix}$$

Παραγοντοποιώντας το γινόμενο των οδηγών παίρνουμε $D = -8A^2(A+3)^2(2A+1)$. Θέτοντας $D = 0$, όπως ορίζει η αναγκαία συνθήκη για την ύπαρξη κοινών μηδενικών και λύνοντας για A παίρνουμε την ίδια λύση με πριν:

$A = -3, -3, -\frac{1}{2}, 0, 0$

Στο επόμενο παράδειγμά μας, η συνθήκη δεν ισχύει. Επομένως, το θεώρημα δεν μπορεί να εφαρμοστεί. Αν όμως εφαρμόσουμε τον αλγόριθμο παίρνουμε τις τιμές της παραμέτρου για κάποιες λύσεις του συστήματος!

3.5 Παράδειγμα 11: Επέκταση του παραδείγματος 4

Είχαμε:

$$M = \begin{bmatrix} 2A^2 + 6A & -2A^2 - 6A \\ -2A^2 - 6A & 2A^2 + 6A \end{bmatrix}$$

Αφού η δεύτερη στήλη του πίνακα Dixon είναι βαθμωτό πολλαπλάσιο της πρώτης, η συνθήκη δεν ισχύει. Άρα, το θεώρημα δεν μπορεί να εφαρμοστεί. Αν πάρουμε την ανηγμένη κλιμακωτή μορφή του πίνακα παίρνουμε:

$$M' = \begin{bmatrix} 2A\left(A+3\right) & -2A\left(A+3\right) \\ 0 & 0 \end{bmatrix}$$

Και από τον οδηγό παίρνουμε τις λύσεις $A = 0, -3$.

Στην πραγματικότητα οι λύσεις του συστήματος είναι:

$(x, A) = (3, -3), (0, 0), (1, A)$

Σε αυτή την περίπτωση πήραμε τις τιμές της παραμέτρου για όλες εκτός από μία λύση. Μόνο η λύση $(1,$A$)$ δεν ανακτήθηκε από το γινόμενο των οδηγών αυτή τη φορά.

3.6 Παράδειγμα 12: Επέκταση του παραδείγματος 8

Είχαμε:

$$M = \begin{bmatrix} 0 & 2A-3 & 0 & 2A-1 & 2A \\ 0 & 2A-3 & 0 & 2A-1 & 2A \\ 2A-3 & -2A^2+3A-2 & 2A-3 & -2A^2+5A-4 & 2A^2 \\ 2A-1 & 2A & 2A-1 & 4A-2 & 2A^2-2A \\ 2A & 2A^3+A^2+A & 2A & 2A^3+A^2+A & 2A^2-2A \end{bmatrix}$$

Πρώτα δείχνουμε ότι η συνθήκη του Θεωρήματος ισχύει, αποδεικνύοντας ότι η τελευταία στήλη του πίνακα Dixon M είναι γραμμικώς ανεξάρτητη από τις υπόλοιπες. Όντως, αν θέσουμε $A = -1$ στον M και πάρουμε την την ανηγμένη κλιμακωτή μορφή του, έχουμε:

$$M_{A=-1} = \begin{bmatrix} 0 & -5 & 0 & -3 & -2 \\ 0 & -5 & 0 & -3 & -2 \\ -5 & -7 & -5 & -11 & 2 \\ -3 & -2 & -3 & -6 & 4 \\ -2 & -2 & -2 & -2 & 4 \end{bmatrix} \text{ και } M' = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Παρατηρούμε ότι η τελευταία γραμμή του M' είναι της μορφής $\begin{bmatrix} 0 & 0 & \dots & 1 \end{bmatrix}$. Οπότε η τελευταία στήλη του M' (άρα και του M) είναι γραμμικώς ανεξάρτητη από τις υπολοίπες. Επομένως, η συνθήκη ισχύει και το Θεώρημα μπορεί να εφαρμοστεί.

Η Dixon-Kapur-Saxen-Yang resultant προκύπτει από το γινόμενο των οδηγών στοιχείων της ανηγμένης κλιμακωτής μορφής του πίνακα Dixon:
$-8A\,(A-1)\,(A+2)\,(2A-1)^2$

Θέτοντας $D = 0$, παίρνουμε: $A = -2, 0, \frac{1}{2}, \frac{1}{2}$

Οι λύσεις του συστήματος $f(x, y, A) = 0, g(x, y, A) = 0, h(x, y, A) = 0$ είναι:

$(x, y, A) = (0, -2, 1), (3, -5, -2), (0, 0, 0), (\frac{1}{2}, -\frac{3}{2}, \frac{1}{2}), (\frac{1}{2}, 0, \frac{1}{2})$

Αυτή τη φορά, όλες οι τιμές της παραμέτρου A εντοπίστηκαν από την D. Ας θυμηθούμε ότι η κλασική Dixon resultant σε αυτή την περίπτωση ήταν ταυτοτικά μηδέν και δεν μας έδινε κάποια πληροφορία για τις λύσεις.

Ας μελετήσουμε ένα τελευταίο παράδειγμα, όπου η συνθήκη δεν ισχύει.

3.7 Παράδειγμα 13

Έστω τα πολυώνυμα f, g, h των μεταβλητών x, y :

$$f(x, y) = xyz$$
$$g(x, y) = x^2 - z^2$$
$$h(x, y) = x + y + z$$

Το πολυώνυμο Dixon είναι:
$(-xz - yz - z^2)\,\alpha^2\beta^0 + (-xz^2 - z^3)\,\alpha^1\beta^0 + yz^3\alpha^0\beta^0$

Ο πίνακας Dixon είναι:

$$M = \begin{bmatrix} -z & -z & -z^2 \\ -z^2 & 0 & -z^3 \\ 0 & z^3 & 0 \end{bmatrix}$$

Και η ορίζουσά του είναι μηδέν. Να σημειώσουμε ότι στην περίπτωση αυτή η συνθήκη δεν ισχύει (η τελευταία στήλη είναι z φορές η πρώτη), οπότε το Θεώρημα δεν μπορεί να εφαρμοστεί. Αν μετατρέψουμε τον πίνακα, παίρνουμε:

$$M' = \begin{bmatrix} 1 & 0 & z \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Η ορίζουσά του είναι $D = -z^3$ και μας δίνει 0 για $z = 0$. Η πραγματική λύση του

συστήματος είναι: $(x, y, z) = (-z, 0, z)$ για κάθε z.

Σε αυτή την περίπτωση μόνο η τετριμμένη λύση εντοπίστηκε.

ΠΑΡΑΡΤΗΜΑ Α

Ο κώδικας της κλάσης DixonResultant

Παρακάτω παραθέτοτνται κομμάτια από δύο αρχεία κώδικα μετά τις προσθήκες αυτής της διπλωματικής εργασίας που πλέον αποτελούν μέρος του πακέτου υπολογιστικής άλγεβρας SymPy.

```
"""
This module contains functions for two multivariate resultants. These
are:
- Dixon's resultant.
- Macaulay's resultant.
Multivariate resultants are used to identify whether a multivariate
system has common roots. That is when the resultant is equal to zero.
"""

from sympy import IndexedBase, Matrix, Mul, Poly
from sympy import rem, prod, degree_list, diag, simplify
from sympy.polys.monomials import itermonomials, monomial_deg
from sympy.polys.orderings import monomial_key
from sympy.polys.polytools import poly_from_expr, total_degree
from sympy.functions.combinatorial.factorials import binomial
from itertools import combinations_with_replacement
from sympy.utilities.exceptions import SymPyDeprecationWarning

class DixonResultant():
    """
    A class for retrieving the Dixon's resultant of a multivariate
    system.
    Examples
    ========
```

```
>>> from sympy.core import symbols
>>> from sympy.polys.multivariate_resultants import DixonResultant
>>> x, y = symbols('x, y')
>>> p = x + y
>>> q = x ** 2 + y ** 3
>>> h = x ** 2 + y
>>> dixon = DixonResultant(variables=[x, y], polynomials=[p, q, h])
>>> poly = dixon.get_dixon_polynomial()
>>> matrix = dixon.get_dixon_matrix(polynomial=poly)
>>> matrix
Matrix([
[ 0,  0, -1,  0, -1],
[ 0, -1,  0, -1,  0],
[-1,  0,  1,  0,  0],
[ 0, -1,  0,  0,  1],
[-1,  0,  0,  1,  0]])
>>> matrix.det()
0
See Also
========
Notebook in examples: sympy/example/notebooks.
References
==========
.. [1] [Kapur1994]_
.. [2] [Palancz08]_
"""

def __init__(self, polynomials, variables):
    """
    A class that takes two lists, a list of polynomials and list of
    variables. Returns the Dixon matrix of the multivariate system.
```

```
        Parameters
        ----------
        polynomials : list of polynomials
            A  list of m n-degree polynomials
        variables: list
            A list of all n variables
        """
        self.polynomials = polynomials
        self.variables = variables

        self.n = len(self.variables)
        self.m = len(self.polynomials)

        a = IndexedBase("alpha")
        # A list of n alpha variables (the replacing variables)
        self.dummy_variables = [a[i] for i in range(self.n)]

        # A list of the d_max of each variable.
        self._max_degrees = [max(degree_list(poly)[i] for poly in self.polynomi
            for i in range(self.n)]

    @property
    def max_degrees(self):
        SymPyDeprecationWarning(feature="max_degrees",
                                issue=17763,
                                deprecated_since_version="1.5").warn()
        return self._max_degrees

    def get_dixon_polynomial(self):
        r"""
        Returns
```

```
        =======
        dixon_polynomial: polynomial
            Dixon's polynomial is calculated as:
            delta = Delta(A) / ((x_1 - a_1) ... (x_n - a_n)) where,
            A =  |p_1(x_1,... x_n), ..., p_n(x_1,... x_n)|
                 |p_1(a_1,... x_n), ..., p_n(a_1,... x_n)|
                 |...              , ...,              ...|
                 |p_1(a_1,... a_n), ..., p_n(a_1,... a_n)|
        """
        if self.m != (self.n + 1):
            raise ValueError('Method invalid for given combination.')

        # First row
        rows = [self.polynomials]

        temp = list(self.variables)

        for idx in range(self.n):
            temp[idx] = self.dummy_variables[idx]
            substitution = {var: t for var, t in zip(self.variables, temp)}
            rows.append([f.subs(substitution) for f in self.polynomials])

        A = Matrix(rows)

        terms = zip(self.variables, self.dummy_variables)
        product_of_differences = Mul(*[a - b for a, b in terms])
        dixon_polynomial = (A.det() / product_of_differences).factor()

        return poly_from_expr(dixon_polynomial, self.dummy_variables)[0]

    def get_upper_degree(self):
```

```
        SymPyDeprecationWarning(feature="get_upper_degree",
                                useinstead="get_max_degrees",
                                issue=17763,
                                deprecated_since_version="1.5").warn()

        list_of_products = [self.variables[i] ** self._max_degrees[i]
                            for i in range(self.n)]
        product = prod(list_of_products)
        product = Poly(product).monoms()

        return monomial_deg(*product)

    def get_max_degrees(self, polynomial):
        r"""
        Returns a list of the maximum degree of each variable appearing
        in the coefficients of the Dixon polynomial. The coefficients are
        viewed as polys in x_1, ... , x_n.
        """
        deg_lists = [degree_list(Poly(poly, self.variables))
                     for poly in polynomial.coeffs()]

        max_degrees = [max(degs) for degs in zip(*deg_lists)]

        return max_degrees

    def get_dixon_matrix(self, polynomial):
        r"""
        Construct the Dixon matrix from the coefficients of polynomial
        \alpha. Each coefficient is viewed as a polynomial of x_1, ...,
        x_n.
        """
```

```
        max_degrees = self.get_max_degrees(polynomial)

        # list of column headers of the Dixon matrix.
        monomials = itermonomials(self.variables, max_degrees)
        monomials = sorted(monomials, reverse=True,
                           key=monomial_key('lex', self.variables))

        dixon_matrix = Matrix([[Poly(c, *self.variables).coeff_monomial(m)
                                for m in monomials]
                                for c in polynomial.coeffs()])

        # remove columns if needed
        if dixon_matrix.shape[0] != dixon_matrix.shape[1]:
            keep = [column for column in range(dixon_matrix.shape[-1])
                    if any([element != 0 for element
                        in dixon_matrix[:, column]])]

            dixon_matrix = dixon_matrix[:, keep]

        return dixon_matrix

    def KSY_precondition(self, matrix):
        """
        Test for the validity of the Kapur-Saxena-Yang precondition.
        The precondition requires that the column corresponding to the
        monomial 1 = x_1 ^ 0 * x_2 ^ 0 * ... * x_n ^ 0 is not a linear
        combination of the remaining ones. In sympy notation this is
        the last column. For the precondition to hold the last non-zero
        row of the rref matrix should be of the form [0, 0, ..., 1].
        """
```

```
        if matrix.is_zero_matrix:
            return False

        m, n = matrix.shape

        # simplify the matrix and keep only its non-zero rows
        matrix = simplify(matrix.rref()[0])
        rows = [i for i in range(m) if any(matrix[i, j] != 0 for j in range(n))
        matrix = matrix[rows,:]

        condition = Matrix([[0]*(n-1) + [1]])

        if matrix[-1,:] == condition:
            return True
        else:
            return False

    def delete_zero_rows_and_columns(self, matrix):
        """Remove the zero rows and columns of the matrix."""
        rows = [
            i for i in range(matrix.rows) if not matrix.row(i).is_zero_matrix]
        cols = [
            j for j in range(matrix.cols) if not matrix.col(j).is_zero_matrix]

        return matrix[rows, cols]

    def product_leading_entries(self, matrix):
        """Calculate the product of the leading entries of the matrix."""
        res = 1
        for row in range(matrix.rows):
            for el in matrix.row(row):
```

```
            if el != 0:
                res = res * el
                break
    return res

def get_KSY_Dixon_resultant(self, matrix):
    """Calculate the Kapur-Saxena-Yang approach to the Dixon Resultant."""
    matrix = self.delete_zero_rows_and_columns(matrix)
    _, U, _ = matrix.LUdecomposition()
    matrix = self.delete_zero_rows_and_columns(simplify(U))

    return self.product_leading_entries(matrix)
```

ΠΑΡΑΡΤΗΜΑ Β

Ο κώδικας για τα tests της κλάσης DixonResultant

```
"""Tests for Dixon's and Macaulay's classes. """

from sympy import Matrix, factor
from sympy.core import symbols
from sympy.tensor.indexed import IndexedBase

from sympy.polys.multivariate_resultants import (DixonResultant,
                                                 MacaulayResultant)

c, d = symbols("a, b")
x, y = symbols("x, y")

p =  c * x + y
q =  x + d * y

dixon = DixonResultant(polynomials=[p, q], variables=[x, y])
macaulay = MacaulayResultant(polynomials=[p, q], variables=[x, y])

def test_dixon_resultant_init():
    """Test init method of DixonResultant."""
    a = IndexedBase("alpha")

    assert dixon.polynomials == [p, q]
    assert dixon.variables == [x, y]
    assert dixon.n == 2
    assert dixon.m == 2
    assert dixon.dummy_variables == [a[0], a[1]]
```

```python
def test_get_dixon_polynomial_numerical():
    """Test Dixon's polynomial for a numerical example."""
    a = IndexedBase("alpha")

    p = x + y
    q = x ** 2 + y **3
    h = x ** 2 + y

    dixon = DixonResultant([p, q, h], [x, y])
    polynomial = -x * y ** 2 * a[0] - x * y ** 2 * a[1] - x * y * a[0] \
    * a[1] - x * y * a[1] ** 2 - x * a[0] * a[1] ** 2 + x * a[0] - \
    y ** 2 * a[0] * a[1] + y ** 2 * a[1] - y * a[0] * a[1] ** 2 + y * \
    a[1] ** 2

    assert dixon.get_dixon_polynomial().factor() == polynomial

def test_get_max_degrees():
    """Tests max degrees function."""

    p = x + y
    q = x ** 2 + y **3
    h = x ** 2 + y

    dixon = DixonResultant(polynomials=[p, q, h], variables=[x, y])
    dixon_polynomial = dixon.get_dixon_polynomial()

    assert dixon.get_max_degrees(dixon_polynomial) == [1, 2]

def test_get_dixon_matrix():
    """Test Dixon's resultant for a numerical example."""
```

```
    x, y = symbols('x, y')

    p = x + y
    q = x ** 2 + y ** 3
    h = x ** 2 + y

    dixon = DixonResultant([p, q, h], [x, y])
    polynomial = dixon.get_dixon_polynomial()

    assert dixon.get_dixon_matrix(polynomial).det() == 0

def test_get_dixon_matrix_example_two():
    """Test Dixon's matrix for example from [Palancz08]_."""
    x, y, z = symbols('x, y, z')

    f = x ** 2 + y ** 2 - 1 + z * 0
    g = x ** 2 + z ** 2 - 1 + y * 0
    h = y ** 2 + z ** 2 - 1

    example_two = DixonResultant([f, g, h], [y, z])
    poly = example_two.get_dixon_polynomial()
    matrix = example_two.get_dixon_matrix(poly)

    expr = 1 - 8 * x ** 2 + 24 * x ** 4 - 32 * x ** 6 + 16 * x ** 8
    assert (matrix.det() - expr).expand() == 0

def test_KSY_precondition():
    """Tests precondition for KSY Resultant."""
    A, B, C = symbols('A, B, C')

    m1 = Matrix([[1, 2, 3],
```

```
                [4, 5, 12],
                [6, 7, 18]])

    m2 = Matrix([[0, C**2],
                [-2 * C, -C ** 2]])

    m3 = Matrix([[1, 0],
                [0, 1]])

    m4 = Matrix([[A**2, 0, 1],
                [A, 1, 1 / A]])

    m5 = Matrix([[5, 1],
                [2, B],
                [0, 1],
                [0, 0]])

    assert dixon.KSY_precondition(m1) == False
    assert dixon.KSY_precondition(m2) == True
    assert dixon.KSY_precondition(m3) == True
    assert dixon.KSY_precondition(m4) == False
    assert dixon.KSY_precondition(m5) == True

def test_delete_zero_rows_and_columns():
    """Tests method for deleting rows and columns containing only zeros."""
    A, B, C = symbols('A, B, C')

    m1 = Matrix([[0, 0],
                [0, 0],
                [1, 2]])
```

```
m2 = Matrix([[0, 1, 2],
             [0, 3, 4],
             [0, 5, 6]])

m3 = Matrix([[0, 0, 0, 0],
             [0, 1, 2, 0],
             [0, 3, 4, 0],
             [0, 0, 0, 0]])

m4 = Matrix([[1, 0, 2],
             [0, 0, 0],
             [3, 0, 4]])

m5 = Matrix([[0, 0, 0, 1],
             [0, 0, 0, 2],
             [0, 0, 0, 3],
             [0, 0, 0, 4]])

m6 = Matrix([[0, 0, A],
             [B, 0, 0],
             [0, 0, C]])

assert dixon.delete_zero_rows_and_columns(m1) == Matrix([[1, 2]])

assert dixon.delete_zero_rows_and_columns(m2) == Matrix([[1, 2],
                                                         [3, 4],
                                                         [5, 6]])

assert dixon.delete_zero_rows_and_columns(m3) == Matrix([[1, 2],
                                                         [3, 4]])
```

```
    assert dixon.delete_zero_rows_and_columns(m4) == Matrix([[1, 2],
                                                             [3, 4]])

    assert dixon.delete_zero_rows_and_columns(m5) == Matrix([[1],
                                                             [2],
                                                             [3],
                                                             [4]])

    assert dixon.delete_zero_rows_and_columns(m6) == Matrix([[0, A],
                                                             [B, 0],
                                                             [0, C]])


def test_product_leading_entries():
    """Tests product of leading entries method."""
    A, B = symbols('A, B')

    m1 = Matrix([[1, 2, 3],
                 [0, 4, 5],
                 [0, 0, 6]])

    m2 = Matrix([[0, 0, 1],
                 [2, 0, 3]])

    m3 = Matrix([[0, 0, 0],
                 [1, 2, 3],
                 [0, 0, 0]])

    m4 = Matrix([[0, 0, A],
                 [1, 2, 3],
                 [B, 0, 0]])
```

```
    assert dixon.product_leading_entries(m1) == 24
    assert dixon.product_leading_entries(m2) == 2
    assert dixon.product_leading_entries(m3) == 1
    assert dixon.product_leading_entries(m4) == A * B

def test_get_KSY_Dixon_resultant_example_one():
    """Tests the KSY Dixon resultant for example one"""
    x, y, z = symbols('x, y, z')

    p = x * y * z
    q = x**2 - z**2
    h = x + y + z
    dixon = DixonResultant([p, q, h], [x, y])
    dixon_poly = dixon.get_dixon_polynomial()
    dixon_matrix = dixon.get_dixon_matrix(dixon_poly)
    D = dixon.get_KSY_Dixon_resultant(dixon_matrix)

    assert D == -z**3

def test_get_KSY_Dixon_resultant_example_two():
    """Tests the KSY Dixon resultant for example two"""
    x, y, A = symbols('x, y, A')

    p = x * y + x * A + x - A**2 - A + y**2 + y
    q = x**2 + x * A - x + x * y + y * A - y
    h = x**2 + x * y + 2 * x - x * A - y * A - 2 * A

    dixon = DixonResultant([p, q, h], [x, y])
    dixon_poly = dixon.get_dixon_polynomial()
    dixon_matrix = dixon.get_dixon_matrix(dixon_poly)
    D = factor(dixon.get_KSY_Dixon_resultant(dixon_matrix))
```

```
assert D == -8*A*(A - 1)*(A + 2)*(2*A - 1)**2
```