JOURNAL de Théorie des Nombres de BORDEAUX

anciennement Séminaire de Théorie des Nombres de Bordeaux

Souad EL OTMANI, Armand MAUL, Georges RHIN et Jean-Marc SAC-ÉPÉE

Integer Linear Programming applied to determining monic hyperbolic irreducible polynomials with integer coefficients and span less than 4

Tome 25, n° 1 (2013), p. 71-78.

<http://jtnb.cedram.org/item?id=JTNB_2013__25_1_71_0>

Journal de Théorie des Nombres
de Bordeaux **25** (2013), 71-78

Integer Linear Programming applied to determining monic hyperbolic irreducible polynomials with integer coefficients and span less than 4

par Souad EL OTMANI, Armand MAUL, Georges RHIN
et Jean-Marc SAC-ÉPÉE

Résumé. Dans ce travail, nous proposons une nouvelle méthode destinée à trouver des polynômes unitaires irréductibles à racines réelles, à coefficients entiers, et dont le diamètre soit inférieur à 4. L'idée principale est de ramener la recherche de tels polynômes à la résolution d'un problème d'optimisation en entiers. Dans ce cadre, les coefficients des polynômes que nous cherchons sont les inconnues entières du problème. Nous donnons des contraintes sur les coefficients induites par les propriétés que l'on s'attend à trouver pour de tels polynômes, notamment une répartition particulière de leurs racines. Ces propriétés s'inspirent de celles des polynômes déjà connus dans la littérature relative à ce domaine.

Abstract. In this work, we propose a new method to find monic irreducible polynomials with integer coefficients, only real roots, and span less than 4. The main idea is to reduce the search of such polynomials to the solution of Integer Linear Programming problems. In this frame, the coefficients of the polynomials we are looking for are the integer unknowns. We give inequality constraints specified by the properties that the polynomials should have, such as the typical distribution of their roots. These properties can be inferred from those of polynomials already treated in the literature on this topic.

1. Introduction

In many situations, it is of interest to evaluate the number of real roots of given polynomials into given intervals. In this paper, we consider one variable monic polynomials which are irreducible over the integers, have integer coefficients and for which all roots are real. For such specific polynomials, important results are available:

Manuscrit reçu le 24 janvier 2012, révisé le 27 juin 2012.

- The set of such polynomials with roots in $[-2.0, 2.0]$ is infinite (Kronecker [6]).
- For each interval with length less than 4, the set of such polynomials with roots inside this interval is finite (Schur [9]).
- For each interval with length greater than 4, the set of such polynomials with roots inside this interval is infinite (Robinson [7]).

If the length of the interval is exactly 4, we do not know whether the set of such polynomials with roots inside this interval is finite or not, except in particular cases as the one mentioned above (Kronecker [6]). Keeping in mind that the *span* of a polynomial with all real roots is merely the difference between its largest and its smallest roots, several interesting articles have been devoted to the study of such polynomials with span lower than 4. Significant results by Robinson ([8]) were completed and developed by Capparelli, Del Fra and Sciò ([1]), who provided a (possibly non exhaustive) list of such polynomials up to degree 17 (three polynomials at degree 17). V. Flammang, G. Rhin and Q. Wu ([5]) proved that this list is exhaustive up to degree 15.

Because of the very wide range of values taken by the coefficients of the already known polynomials, it would be absolutely impossible to directly find new valid polynomials by varying the coefficients into large intervals, as the number of polynomials that we should test would be prohibitive. It seems therefore appropriate to tackle the problem of finding polynomials with higher degrees by a completely alternative approach, based on Integer Linear Programming methods. The basic idea is to look for polynomials with small spans by looking for polynomials having all their roots inside intervals with small lengths.

2. Search interval

2.1. Reducing searches to the interval $(-2, 2.5)$. First of all, notice that if an integer polynomial has only real roots and span less than 4, then (by an integral translation) one can always obtain this polynomial from an other polynomial with roots inside the interval $(-2, 3)$. For example, the polynomial $x^3-15x^2+72x-109$, with roots $3.120614\cdots$, $5.347296\cdots$ and $6.532088\cdots$, can be obtained from the polynomial x^3-3x+1, whose roots $-1.879385\cdots$, $0.347296\cdots$ and $1.532088\cdots$ are in the interval $(-2, 3)$, via a simple translation since $x^3 - 15x^2 + 72x - 109 = (x-5)^3 - 3(x-5) + 1$.

Now, since we are interested in polynomials with span less than 4, the search interval can be reduced again from $(-2, 3)$ to $(-2, 2.5)$. Indeed, let us consider a polynomial P with span less than 4, all the roots of which are in $(-2, 3)$ and with at least one root between 2.5 and 3. If we take the opposites of all its roots, we translate these values to the interval between

-2 and 2.5 via an integral translation, then we obtain a polynomial with the same span as P.

2.2. Splitting of the interval $(-2, 2.5)$ into smaller subintervals. As we shall see in section 3, our method allows us to search for integer polynomials with all roots contained in a given interval. Of course, it would not be effective to directly search for polynomials with all roots into the interval $(-2, 2.5)$, because we would mostly obtain polynomials with too large spans (polynomials are more and more numerous as their spans increase).

Then, an effective idea is to split the interval $(-2, 2.5)$ into several smaller intervals all included in $(-2, 2.5)$, with spans slightly larger than 4, and to apply our method to each of them. Indeed, if we look for an integer polynomial with all roots in an interval with span slightly larger than 4, say $4 + \varepsilon$, then the possibly obtained polynomials necessarily have spans less than $4 + \varepsilon$ (and then maybe less than 4).

Let us also remark that, even by using intervals with span close to 4, we will have to verify the spans of the polynomials we obtain. For example, the polynomial $p = x^{18} - 6x^{17} - x^{16} + 65x^{15} - 63x^{14} - 292x^{13} + 406x^{12} + 709x^{11} - 1137x^{10} - 1021x^9 + 1698x^8 + 902x^7 - 1379x^6 - 483x^5 + 563x^4 + 143x^3 - 90x^2 - 17x + 1$ was found by our algorithm: it is not valid because its span is 4.0000610483949. Furthermore, we will have to verify that the polynomials we find are irreducible.

3. Formulation as an optimization problem

3.1. Presentation of the optimization problem formulation. In this section, we describe how to find a one-variable monic polynomial with integer coefficients, only real roots, and with all roots between two fixed real numbers. Of course, our purpose is to apply this method to fixed subintervals of $(-2, 2.5)$.

Recall that a hyperbolic polynomial is a one-variable polynomial all roots of which are real. Suppose that we want to find a monic hyperbolic polynomial of degree d, with integer coefficients and for which every root lies between two fixed real values a and b.

Set $p(x) = x^d + \sum_{i=0}^{d-1} a_i x^i$, where the coefficients a_0, a_1, …, a_{d-1} are integers. Suppose this polynomial has its roots $\alpha_1, \ldots, \alpha_d$ between a and b. Then, for each $(\beta_1, \ldots, \beta_{d-1})$ which separates the roots of the polynomial, one necessarily has either $p(a) > 0$, $p(\beta_1) < 0$, $p(\beta_2) > 0, \ldots$, $p(\beta_{d-1}) < 0$, $p(b) > 0$, if p is an even-degree polynomial, or one has $p(a) < 0$, $p(\beta_1) > 0$, $p(\beta_2) < 0, \ldots$, $p(\beta_{d-1}) < 0$, $p(b) > 0$ otherwise.

Then, for a given $(\beta_1, \ldots, \beta_{d-1})$, consider the following linear programming problem where all the unknowns (i.e. the a_i's) are restricted to integer values:

$$\text{Minimize } \sum_{i=0}^{d-1} a_i,$$
$$\text{subject to } \begin{cases} (-1)^d p(a) > 0, \\ (-1)^d p(\beta_1) < 0, \\ \vdots \\ p(\beta_{d-1}) < 0, \\ p(b) > 0, \\ (a_0, \cdots, a_{d-1}) \in \mathbb{Z}^d. \end{cases} \tag{3.1}$$

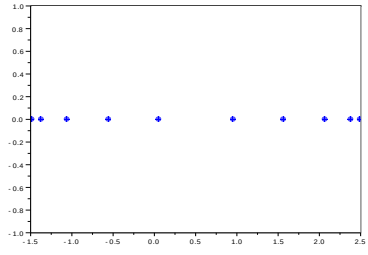
In the calculations, the values a, b (the end-points of each subinterval of $(-2, 2.5)$) and the β_i's (which we will fix in a specific way via samples from a suitable random distribution) are known, while the a_i's are the unknowns of the problem.

Notice that the linear expression we wish to minimize is not very important, since above all we want to find integers $a_0, \ldots, a_{d-1}$ which satisfy the inequality constraints.

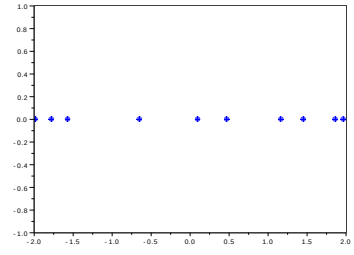
Now, the existence (on each subinterval) of valid polynomials satisfying our system depends on the values of the β_i's. It is therefore extremely important to use "well-placed"β_i's if we want to have a chance to find new polynomials.

3.2. Distribution of the β_i's. A natural idea to select the β_i's is to have a close look at the distribution of roots arising from polynomials already known to satisfy all requirements.

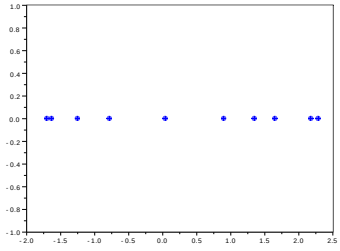
The following figures (see Capparelli, Del Fra and Sciò [1]) show the roots distribution of some polynomials (of degree 10). The integers under each graph indicate the coefficients of the polynomials.

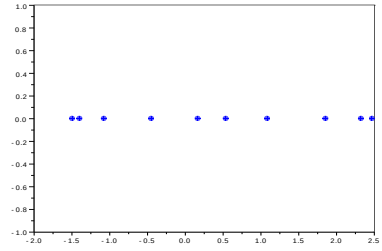


1 -5 0 30 -20 -66 46 63 -28 -21 1



1 -1 -10 10 34 -34 -43 43 12 -12 1





1 -3 -7 24 16 -67 -12 75 0 -27 1 1 -4 -3 24 -2 -49 11 36 -8 -6 1

Through these examples, we notice that there is manifestly a specific distribution of roots, which we wish to exploit to guess how the polynomials of higher degrees behave. As we can see, roots are very close together when they are in the neighbourhood of the end-points of the intervals while they are more and more spread as they lie near the center of the intervals.

This pattern can in fact be verified in every available polynomial. So, let us now come back to our work at degree 16 and higher. Real values arising from a beta distribution with appropriate parameters typically spread out in this way. Then, we computed the set of roots of each already known polynomial, and for each set of roots, we estimated the parameters of the corresponding beta distribution. A beta distribution is characterized by two shape parameters, but in our particular case, the arrangement of roots led us to suppose that the two parameters are equal.

We used the method of moments to estimate this parameter for each known polynomial at degrees 16 and 17, and we noticed that in every case, the value of this estimated parameter is close to 0.48.

Then, the idea of our method is as follows: for each subinterval, we generate numerous samples (distributed like roots of the known polynomials) stemming from a beta distribution with equal parameters close to 0.48, and we choose as β_i's the middles of these values. Then, for each such $(\beta_1, \ldots, \beta_n)$, we solve a system similar to system (1). The resolution can fail if no $(a_0, \ldots, a_n)$ suits. Otherwise, either the resolution can supply some $(a_0, \ldots, a_n)$ which corresponds to a not irreducible polynomial, or a satisfying $(a_0, \ldots, a_n)$ can be found.

For the reader's convenience, we outline some Integer Linear Programming theory in the following section.

4. A brief outline of Linear Integer Programming

A Linear Integer Programming model seeks to optimize a linear objective function, subject to some equalities and inequalities. More precisely, such a problem writes :

$$\max_x (\text{or} \min_x)\ f(x) \text{ subject to } C(x),$$

where $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is a linear function, and $C(x)$ is a conjunction of linear equality and inequality constraints. Moreover, the unknowns x_i are required to have integer values.

For a pure Integer Linear Problem (as system (1)), an efficient approach is to use the branch-and-bound algorithm together with Gomory's mixed integer cuts.

The branch-and-bound algorithm enumerates all candidate solutions, and rules out large quantities of unfit candidates by using upper and lower estimated bounds of the objective value. Gomory's method consists of ignoring that the x_i's are supposed to be integer values, and first solving an associated linear programming problem. If the solution found is not an integer, one introduces an additional constraint (the so-called Gomory's constraint) corresponding to the condition for integrability. This condition excludes numerous non integer solutions but preserves integral ones. Then, a usual simplex method is used to solve this new problem. If the solution found is still not an integer, then we add another new constraint and solve again. The process is repeated until obtaining an integer solution.

5. Implementation and results

5.1. Implementation. Since it is sufficient to search for polynomials with span less than 4 and with all roots into the interval $[-2.0, 2.5]$ to handle all the possible cases (see section 2), we partitioned this interval into 40 slices (of length 4.0125): $[-2.0, 2.0125]$, $[-1.9875, 2.025]$, ..., $[-1.5125, 2.5]$. This choice of the number of slices was simply based on the number of available systems: 5 dual-core quad-processor systems.

The interest in using these small subintervals is to avoid the numerous polynomials with spans higher than 4.0125.

To handle the 40 subintervals at the same time, we have developed a C++ program based on the MPI library ([10]), which we run on parallel computers available for this type of calculations. The program implements the algorithm explained in subsection 3.2. On each subinterval of $[-2.0, 2.5]$, each loop has three steps:

(1) **Drawing** a sample for the β_i's,
(2) **Solving** the linear programming problem,
(3) **Testing** the obtained polynomial (span and irreducibility).

The process is iterated by requiring each time a new sample for the β_i's until a stop condition occurs. Of course, step 1 and step 3 are much faster than step 2.

The calculations of the roots of the polynomials (for the testing) are made via GSL library ([4]), while the tests of irreducibility are made via the GP-PARI numbers theory library ([2]). Solving Linear Integer Programming systems on each subintervals is processed thanks to the GLPK library ([3]).

All the calculations are made through a single C++ program who calls the aforesaid libraries to achieve the various required tasks at each step of the process.

Notice that we do not need a great accuracy for the localization of the β_i's: if some $(\beta_1, \cdots, \beta_n)$ separates the roots of a polynomial, another $(\beta'_1, \cdots, \beta'_n)$ close to the first one also separates the roots of this polynomial.

We decided to stop the calculations when the same polynomials are repeatedly found, and when new polynomials (valid or not) do not appear anymore.

We let the program run for a day for degree 16, and we doubled the duration at each increase of degree. In fact, we noticed that it was possible to stop the program much earlier, because the resulting polynomials (valid or not) appeared quickly, before being repeated again and again.

For reducible polynomials found at degrees 17, 18, 19 and 20, we also verified their factors because some of them could have been a valid polynomial for lower degrees.

5.2. Results. Since the lists available in the article by Capparelli, Del Fra and Sciò ([1]) are not proved to be exhaustive at degrees 16 and 17, we first tried to search for new polynomials at these degrees. In fact, although no proof for this is available, we are convinced that they are indeed complete, as our calculations at degrees 16 and 17 allowed us to find all the polynomials given in [1], and no new ones. For the interval $[-2.0, 2.0125]$, we have forced the polynomials to have a root between 2.0 and 2.0125 to automatically eliminate polynomials of cosine type, which have all roots between -2.0 and 2.0.

Of course, the subsequent step was to search for new polynomials, still unknown. We thus tested the efficiency of our algorithm at degree 18. The following three new polynomials, given below with their spans, were found by using our algorithm:

- $x^{18} - 2x^{17} - 16x^{16} + 31x^{15} + 107x^{14} - 198x^{13} - 388x^{12} + 672x^{11} + 827x^{10} - 1302x^9 - 1048x^8 + 1436x^7 + 758x^6 - 844x^5 - 280x^4 + 225x^3 + 40x^2 - 19x - 1$ of span 3.9760414
- $x^{18} - 8x^{17} + 12x^{16} + 58x^{15} - 174x^{14} - 116x^{13} + 770x^{12} - 108x^{11} - 1702x^{10} + 734x^9 + 2095x^8 - 1065x^7 - 1440x^6 + 637x^5 + 496x^4 - 138x^3 - 56x^2 + 8x + 1$ of span 3.9955284
- $x^{18} - 6x^{17} - x^{16} + 66x^{15} - 68x^{14} - 293x^{13} + 444x^{12} + 678x^{11} - 1242x^{10} - 891x^9 + 1816x^8 + 695x^7 - 1396x^6 - 333x^5 + 500x^4 + 89x^3 - 56x^2 - 5x + 1$ of span 3.9968482

6. conclusion

In spite of similar calculations made at degrees 19 and 20, new irreducible polynomials did not appear at these degrees, but polynomials available in

Capparelli, Del Fra and Sciò ([1]) and our three polynomials above appeared again as factors of numerous not irreducible polynomials supplied by our program at these degrees. In the even higher degrees, problems of convergence appear with GLPK. Perhaps, it will be necessary to test alternative optimization libraries to solve our Integer Linear Programming systems.

References

[1] S. Capparelli, A. Del Fra, C. Sciò, *On the span of polynomials with integer coefficients.* Mathematics of Computation, S 0025-5718(09)02292-3.
[2] *PARI/GP, version* `2.5.0`. Bordeaux, 2011, `http://pari.math.u-bordeaux.fr/`
[3] *GNU Linear Programming Kit, version 4.35.* `http://www.gnu.org/software/glpk/glpk.html`
[4] M. Galassi et al, *GNU Scientific Library Reference Manual (2nd Ed.)*. ISBN 0954161734.
[5] V. Flammang, G. Rhin, Q. Wu, *The Totally Real Algebraic Integers with Diameter less than 4.* Moscow Journal of Combinatorics and Number Theory Vol. **1** (2011), Iss. 1, 21–32.
[6] L. Kronecker, *Zwei Sätze über Gleichungen mit ganzzahligen Koefficienten.* J. Reine Angew. Math. **53** (1857), 173–175.
[7] R. Robinson, *Intervals containing infinitely many sets of conjugate algebraic integers.* Studies in Mathematical Analysis and Related Topics: Essays in honor of George Pólya, Stanford Univ. Press, 1962, 305-315. MR0144892 (26:2433).
[8] R. Robinson, *Algebraic equations with span less than 4.* Math. Comp. **18** (1964), 547–559. MR0169374 (29:6624).
[9] I. Schur, *Über die Verteilung der Würzeln bei gewissen algebraischen Gleichungen mit ganzzahligen Koeffizienten.* Math. Z. **1** (1918), 377–402. MR1544303.
[10] *MPI Forum. Message Passing Interface (MPI) Forum Home Page.* `http://www.mpi-forum.org/` (Dec. 2009).

Souad El Otmani
Université de Lorraine, site de Metz
Ile du Saulcy
57045 Metz Cedex
E-mail: `souad.elotmani@univ-lorraine.fr`

Armand Maul
Université de Lorraine, site de Metz
Ile du Saulcy
57050 Metz Cedex
E-mail: `armand.maul@univ-lorraine.fr`
URL: `http://www.math.univ-metz.fr/~maul`

Georges Rhin
Université de Lorraine, site de Metz
Ile du Saulcy
57050 Metz Cedex
E-mail: `georges.rhin@univ-lorraine.fr`
URL: `http://www.math.univ-metz.fr/~rhin`

Jean-Marc Sac-Épée
Université de Lorraine, site de Metz
Ile du Saulcy
57050 Metz Cedex
E-mail: `jean-marc.sac-epee@univ-lorraine.fr`
URL: `http://www.math.univ-metz.fr/~jmse`