Gröbner basis solutions to satellite trajectory control by pole placement

Zuzana Kukelova, Pavel Krsek, Vladimir Smutny, Tomas Pajdla
Center for Machine Perception
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Czech Republic
{kukelova,krsek,smutny,pajdla}@cmp.felk.cvut.cz

ABSTRACT

Controlling satellite trajectories is an important problem. In [12], an approach to the pole placement for the synthesis of a linear controller has been presented. It leads to solving five polynomial equations in nine unknown elements of the state space matrices of a compensator. This is an underconstrained system and therefore four of the unknown elements need to be considered as free parameters and set to some prior values to obtain a system of five equations in five unknowns. In [12], this system was solved for one chosen set of free parameters by Dixon resultants. In this work, we study and present Gröbner basis solutions to this problem of computation of a dynamic compensator for the satellite for different combinations of free input parameters. We show that the Gröbner basis method for solving systems of polynomial equations leads to very simple solutions for all combinations of free parameters. These solutions require to perform only the Gauss-Jordan elimination of a small matrix and computation of roots of a single variable polynomial. The maximum degree of this polynomial is not greater than six in general but for most combinations of the input free parameters its degree is even lower.

1. INTRODUCTION

Satellites play an important role, e.g., in telecommunication, navigation and weather monitoring. For the proper operation of satellites, it is necessary to keep them on their orbit and pointed in the right direction. However, satellites may change their orientation due to some outdoor disturbances such as aerodynamic drag, the gravitation acceleration of the sun and moon, or due to internal disturbances such as movement of mechanical parts of satellites. Therefore, controlling the trajectories of satellites is an important problem.

For this purpose the position and the motion of the satellite are controlled by a programmed control loop consisting of sensors measuring the position and orientation of the satellite. Based on these measurements and according to the control law of the controller commands which influence the satellite's attitude are generated.

A relatively simple dynamic model of satellite motion is sufficient for control design [8, 9]. In this model, polar coordinates are used for describing the satellite position on equatorial orbit and the satellite is controlled via tangential thruster [7]. Such a model leads to a linear dynamic control system. There are many different control design techniques for linear dynamic control systems. One simple technique is the pole placement technique [1, 3, 14] which results in an underconstrained system of polynomial equations. There exist many numerical methods for solving the pole placement problem, such as Ackermann's formula or QR algorithm [6]. The numerical methods are implemented in control design packages of Matlab or Mathematica. However, these numerical methods may suffer from numerical instability for ill-conditioned systems or poles with multiplicity.

In [17], the pole placement problem was solved using the symbolic-numeric algorithm based on Pieri homotopies and with the help of the polynomial homotopy continuation package PHCpack [15]. In [17, 16] the presented homotopy method was also used to solve the satellite trajectory control problem for one concrete combination of parameters.

In [12, 13] the problem of controlling the satellite trajectory by pole placement was solved using a symbolic method based on resultants. The pole placement technique leads for the satellite trajectory control problem to five polynomial equations in nine unknowns. Therefore four unknowns have to be considered as free. In [12, 13] authors solve the resulting system of polynomial equations with the help of Mathematica for one set of free input parameters. They also derive a constraint on these four free parameters which can ensure only real solutions. For this purpose Dixon resultant implemented in Mathematica is used.

In this work, we study and present Gröbner basis solutions to the problem of computation of a dynamic compensator for the satellite for different combinations of free input parameters. We show that the Gröbner basis method for solving systems of polynomial equations leads to simple solutions for all combinations of free parameters. To solve for a large number of problems we use the automatic generator of efficient Gröbner basis solvers presented in [10].

The presented Gröbner basis solutions require to perform only the Gauss-Jordan elimination of a small matrix and computation of roots of a single variable polynomial. The maximum degree of this polynomial is not greater than six in general but for most combinations of the input free parameters its degree is even lower. For some combinations of free

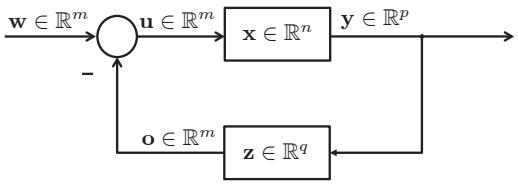

Fig. 1. System with dynamic output feedback

input parameters the resulting solver returns only one solution which ensures, for reasonable inputs, a real solution. All created Gröbner basis solvers are very fast and they return solutions to all variables in few tens of microseconds.

Our approach is particularly useful for testing different parameters of the compensator and for developing numerically stable compensators, when, e.g., state space needs to be transformed to a different coordinate system.

2. Problem formulation

We follow the problem formulation from [12]. Consider a linear system with m inputs $\mathbf{u} \in \mathbb{R}^m$ and p outputs $\mathbf{y} \in \mathbb{R}^p$, given by three matrices $\mathtt{A} \in \mathbb{R}^{n\times n}, \mathtt{B} \in \mathbb{R}^{n\times m}$, and $\mathtt{C} \in \mathbb{R}^{p\times n}$, where n is the number of internal states stored by the vector $\mathbf{x} \in \mathbb{R}^n$. This system is described by linear first order differential equations

$$\dot{\mathbf{x}} = \mathtt{A}\mathbf{x} + \mathtt{B}\mathbf{u}, \tag{1}$$

$$\mathbf{y} = \mathtt{C}\mathbf{x}. \tag{2}$$

Assume that we want to control this system using the dynamic compensator that has q internal states $\mathbf{z} \in \mathbb{R}^q$. The dynamic compensator, see Figure 1, which realizes a feedback law, can be described by the first order differential equations

$$\dot{\mathbf{z}} = \mathtt{F}\mathbf{z} + \mathtt{G}\mathbf{y}, \tag{3}$$

$$\mathbf{o} = \mathtt{H}\mathbf{z} + \mathtt{K}\mathbf{y}, \tag{4}$$

where $\mathtt{F} \in \mathbb{R}^{q\times q}, \mathtt{G} \in \mathbb{R}^{q\times p}, \mathtt{H} \in \mathbb{R}^{m\times q}$, and $\mathtt{K} \in \mathbb{R}^{m\times p}$. Assuming $\mathbf{w} = 0$, we get the close-loop system

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} \mathtt{A} - \mathtt{BKC} & -\mathtt{BH} \\ \mathtt{GC} & \mathtt{F} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix}. \tag{5}$$

There are many different methods for solving the linear control problem. One of the simplest methods is the output feedback pole placement technique. In this technique, given a list of eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{n+q}$ we are looking for laws to feed the output back to the input so that the resulting closed-loop system (5) has the eigenvalues from this list.

The characteristic polynomial of the close-loop matrix from equation (5), i.e. the polynomial whose roots are precisely the eigenvalues of this matrix, has the form

$$q(s) = \det \begin{bmatrix} s\mathtt{I}_n - \mathtt{A} + \mathtt{BKC} & \mathtt{BH} \\ -\mathtt{GC} & s\mathtt{I}_q - \mathtt{F} \end{bmatrix} = 0. \tag{6}$$

This means that in the pole placement technique, given system matrices A,B,C, the state space matrices F,G,H,K of the compensator have to be determined such that the roots of the characteristic polynomial $q(s)$ (6) are equal to the given values (poles) $\lambda_1, \lambda_2, \ldots, \lambda_{n+q}$.

Therefore, the pole placement technique leads to a system of $n+q$ polynomial equations in $(m+q)(p+q)$ unknowns.

2..1 Dynamic compensator for a satellite

The satellite motion can be described by the dynamic model [8] in which polar coordinates are used for describing satellite position in a circular, equatorial orbit. The goal of the feedback is to keep the satellite in this equatorial orbit when disturbances appear.

In dynamic model of the satellite motion, the state vector is

$$\mathbf{x} = \begin{bmatrix} r & \dot{r} & \theta & \dot{\theta} \end{bmatrix}^\top, \tag{7}$$

where r and θ are the deviations from the reference orbit and the reference attitude.

The input is

$$\mathbf{u} = \begin{bmatrix} u_r & u_t \end{bmatrix}^\top, \tag{8}$$

where u_t and u_r are radial and tangential thrusters.

The linear state-space system (1), (2) for the satellite can be then defined by the matrices

$$\mathtt{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 3\omega_0^2 & 0 & 0 & 2\omega_0 r_0 \\ 0 & 0 & 0 & 1 \\ 0 & -2\frac{\omega_0}{r_0} & 0 & 0 \end{bmatrix}, \tag{9}$$

$$\mathtt{B} = \begin{bmatrix} 0 & 0 \\ \frac{1}{v} & 0 \\ 0 & 0 \\ 0 & \frac{1}{v\, r_0} \end{bmatrix}, \tag{10}$$

where v is the mass of the satellite, r_0 the radius and ω_0 the angular velocity.

In [7], it was shown that the satellite is completely controllable with the tangential thruster u_t only. Therefore it is possible to choose $\mathtt{C}$ from (2) as

$$\mathtt{C} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{11}$$

In this case we have $n = 4, m = 2$ and $p = 2$. If we set $q = 1$, then in the pole placement technique we have to assign $n + q = 4 + 1 = 5$ poles $\lambda_1, \ldots, \lambda_5$.

After denoting the state matrices of the dynamic compensator (3), (4) for the satellite as

$$\mathtt{F} = \begin{bmatrix} f_{11} \end{bmatrix}, \mathtt{G} = \begin{bmatrix} g_{11} & g_{12} \end{bmatrix}, \mathtt{H} = \begin{bmatrix} h_{11} \\ h_{21} \end{bmatrix}, \mathtt{K} = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}, \tag{12}$$

the characteristic polynomial $q(s)$ (6) of the close-loop matrix has the form

$$q(s) = \det \begin{bmatrix} s\mathtt{I}_4 - \mathtt{A} + \mathtt{BKC} & \mathtt{BH} \\ -\mathtt{GC} & s\mathtt{I}_1 - \mathtt{F} \end{bmatrix} = \det \begin{bmatrix} s & -1 & 0 & 0 & 0 \\ -3\omega_0^2 & s & \frac{k_{11}}{v} & -2r_0\omega_0 + \frac{k_{12}}{v} & \frac{h_{11}}{v} \\ 0 & 0 & s & -1 & 0 \\ 0 & 2\frac{\omega_0}{r_0} & \frac{k_{21}}{vr_0} & s + \frac{k_{22}}{vr_0} & \frac{h_{21}}{vr_0} \\ 0 & 0 & -g_{11} & -g_{12} & s - f_{11} \end{bmatrix}. \tag{13}$$

After substituting five given eigenvalues (poles) $\lambda_1, \ldots, \lambda_5$, to the characteristic polynomial (13), we obtain five polynomial equations in nine variables $f_{11}, g_{11}, g_{12}, h_{11}, h_{21}, k_{11}, k_{12}, k_{21}, k_{22}$ of the form

$$\begin{aligned} q(\lambda_i) \;=\; & \lambda_i^5 vr_0 + \lambda_i^3 vr_0\omega_0^2 - \lambda_i^4 vr_0 f_{11} - \lambda_i^2 vr_0\omega_0^2 f_{11} - 2\lambda_i\omega_0 h_{11}g_{11} - \lambda_i^2 h_{21}g_{11} - 3\omega_0^2 h_{21}g_{11} - \\ - \;& 2\lambda_i^2\omega_0 h_{11}g_{12} + +\lambda_i^3 h_{21}g_{12} - 3\lambda_i\omega_0^2 h_{21}g_{12} - 2\lambda_i^2\omega_0 k_{11} + 2\lambda_i\omega_0 f_{11}k_{11} - 2\lambda_i^3\omega_0 k_{12} + \\ + \;& 2\lambda_i^2\omega_0 f_{11}k_{12} + \lambda_i^3 k_{21} - 3\lambda_i\omega_0^2 k_{21} - \lambda_i^2 f_{11}k_{21} + 3\omega_0^2 f_{11}k_{21} + \lambda_i^4 k_{22} - 3\lambda_i^2\omega_0^2 k_{22} - \\ - \;& \lambda_i^3 f_{11}k_{22} + 3\lambda_i\omega_0^2 f_{11}k_{22}. \end{aligned} \tag{14}$$

This is an underconstrained system of polynomial equations. Therefore, to have finitely many solutions, four from the nine variables have to be set to some values resulting in five equations in five unknowns of degree five. Next we show how such systems of polynomial equations can be solved using the algebraic Gröbner basis method.

3. Gröbner basis method

The Gröbner basis method for solving systems of polynomial equations has recently became popular in computer vision and robotics and it has been used to create very fast, efficient and numerically stable solvers to many difficult problems. The method is based on polynomial ideal theory and is concerned with special bases of these ideals called Gröbner bases [5]. Gröbner bases have the same solutions as the initial system of polynomial equations defining the ideal but are often easier to solve. Gröbner bases are usually used to construct special multiplication matrices [5], which can be viewed as a generalization of the companion matrix used in solving one polynomial equation in one unknown. The solutions to the system of polynomial equations are then obtained from the eigenvalues and eigenvectors of such multiplication matrices. See [5] for more on Gröbner basis methods and [10, 11, 4, 2] for their applications. Since general algorithms [5] for computing Gröbner bases are not very efficient for solving problems which appear, for example, in computer vision, an automatic generator of specific polynomial equations solvers based on the Gröbner basis method has been proposed in [10]. These specific solvers often provide very efficient solutions to a class of systems of polynomial equations consisting of the same monomials and differing only in the coefficients. Many problems including the problem

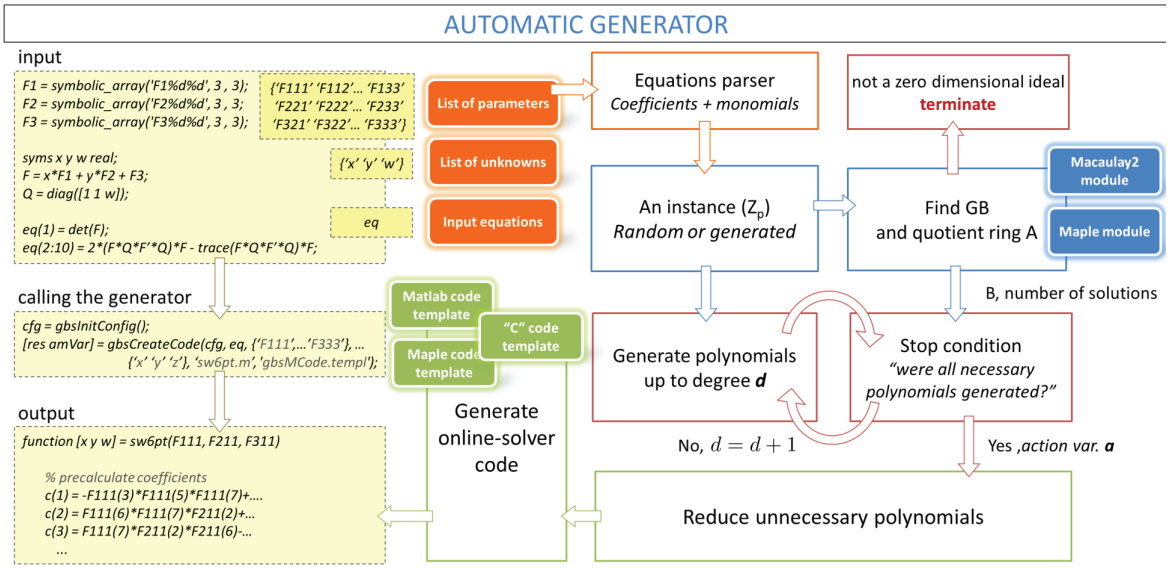

Fig. 2. Illustration of the basic steps of the automatic generator

presented in this paper share the convenient property that the monomials appearing in the set of polynomials resulting from this problem, in this case system (14), are always the same irrespective of the concrete coefficients arising, usually, from non-degenerate measurements. Therefore, it is possible to use efficient specific solvers instead of less efficient general algorithms [5] for computing the Gröbner bases.

4. Automatic generator

Since the Gröbner basis method for solving systems of polynomial equations requires non-trivial knowledge of algebraic geometry from its user, we have proposed an automatic generator of specific Gröbner basis solvers [10] which could be used even by non-experts to easily solve problems leading to systems of polynomial equations. The process of creating these specific Gröbner basis solvers consists of two phases. In the first "offline" phase, the so-called "elimination templates" are found. These templates specify the elimination sequence in order to obtain all polynomials from the Gröbner basis or at least all polynomials necessary for the construction of a multiplication matrix. This phase is performed only once for a given problem. In the second "online" phase, the elimination templates are used with coefficients arising from specific measurements to construct the multiplication matrix. Then, eigenvalues and eigenvectors of the multiplication matrix provide solutions to the original polynomial equations.

Our automatic generator presented in [10] performs the offline phase automatically and for an input system of polynomial equations outputs an efficient online solver. The input into our automatic generator is the system of polynomial equations which we want to solve with a particular choice of coefficients from $\mathbb{Z}_p$ that choose the particular elimination template. For many problems, the interesting "regular" solutions can be obtained with almost any random choice of the coefficients. Therefore, we use random coefficients from $\mathbb{Z}_p$. The output from the generator is the Matlab or C code solver that returns solutions to the input system of polynomial equations for arbitrary "non-degenerate" coefficients from $\mathbb{Q}$. In online computations, only this generated solver is called. The illustration of the basic steps of the automatic generator are in Figure 2.

We have used the automatic generator to create fast and efficient solvers to the problem of computation of a dynamic compensator for the satellite for different combinations of input free parameters.

5. Gröbner basis solutions to the satellite trajectory control

In section 2.1 we have shown that the problem of the computation of the dynamic compensator for the satellite using the pole placement technique results in five polynomial equation in nine unknowns of the form (14). To obtain a finite number of solutions to this system of five equations, four from the nine variables have to be considered as known and set to some prior values.

In [12] the problem of computation of the dynamic compensator for the satellite was solved using Dixon resultants for one combination of free parameters. In this case g_{12}, h_{21}, k_{12} and k_{21} were considered as free parameters and $f_{11}, g_{11}, h_{11}, k_{11}$ and k_{22} as unknowns.

In this section we will show that the automatic generator can be easily used to create fast end efficient Gröbner basis solvers to all combinations of free parameters in our problem. These solutions require to perform only the Gauss-Jordan elimination of a small matrix and computation of roots of a single variable polynomial. The maximum degree of this polynomial is not greater than six in general but for most combinations of the input free parameters its degree is even lower.

If we fix f_{11} as an unknown, then there are 70 different combinations of free parameters and unknowns. Using the automatic generator we have generated 68 different Gröbner basis solvers for all these combinations of free parameters. Two combinations of free parameters, i.e. $k_{11}, k_{12}, k_{21}, k_{22}$ and $h_{11}, h_{21}, k_{12}, k_{22}$ as free, resulted in overconstrained systems of polynomial equations and therefore didn't return results. All the remaining solvers return from 1 to 6 solutions to the unknown parameters. The detailed analysis of all 68 solvers with the number of solutions and the size of the resulting Gauss-Jordan elimination can be found in Table I.

The "size" of the solver (i.e. of the Gauss-Jordan elimination) depends on the monomial ordering used. In all our solvers, we have used the graded reverse lexicographic ordering with the ordering of the unknowns as listed in Table I.

Thanks to the automatic generator, all 68 solvers from Table I can be easily generated in a few minutes. Once we have generated necessary solvers using the automatic generator, we do not need to call it again and we can use the generated fast efficient solvers for arbitrary values of free parameters to get solutions to all unknowns in a few microseconds. The input to the automatic generator [10] can be found in Figure 3.

Note that for some solvers the result from the automatic generator [10] returns solutions only to some from the unknown variables and the solutions to the remaining variables have to be computed using the obtained solutions. Usually the solutions to the remaining variables can be easily extracted from the eliminated coefficient matrix (matrix after Gauss-Jordan elimination) computed in the solver or by solving system of a small number of linear equations.

The solver 20 from Table I solves exactly the combination of free parameters presented in paper [12]. Our Gröbner basis solver to this combination of free parameters leads to one Gauss-Jordan elimination of a 5×7 matrix, computation of roots of one quadratic equation and solving two linear equations in two unknowns.

6. Results

To demonstrate the functionality of our Gröbner basis solutions we have made several experiments. In the first experiment we have tested the Gröbner basis solver 20 from Table I on data from [17, 12, 13] and compared the results of our Gröbner basis solver with the results of the solution presented in [12] and implemented in Mathematica. In this case the values of five poles λ_i in (14) were set to

$$\lambda_1 = \frac{-2+i}{5},\ \lambda_2 = \frac{-2-i}{5},\ \lambda_3 = -5,\ \lambda_4 = -7,\ \lambda_5 = -3, \tag{15}$$

and the parameters of the state-space model of the satellite in (9) and (10) to

$$m = 0.74564,\ \omega_0 = 0.345354,\ r_0 = 1.2342. \tag{16}$$

These values were chosen in [17] randomly and have no real interpretation. However these values were used also in [12, 13] and therefore we chose them for our first experiment to demonstrate that our solver works correctly and returns the same results as the solver from [12, 13].

In [12, 13] the four free parameters were set to

$$l_{12} = 1,\ h_{21} = 1,\ k_{12} = 1,\ k_{21} = 26. \tag{17}$$

Table II shows the results of the Dixon resultant solver presented in [12] and implemented in Mathematica and the results of our Gröbner basis solver generated using the automatic generator [10] and implemented in Matlab.

The results returned from both solvers are the same up to 11^{th} decimal place. Since both solvers are implemented in different programing languages it is not completely fair to compare their running times, however, the running the new Gröbner basis solver is several hundreds times faster than the Dixon resultant solver [12] from Mathematica. Moreover, the Gröbner basis solver can be easily generated using the automatic generator [10] and since it consists only of the Gauss-Jordan elimination of the initial polynomial equations and computation of the root of quadratic polynomial it can be easily reimplemented in another programing language. On the other hand the Dixon resultant solver [12] uses internal commands of Mathematica, such as `DixonResultant` and `Nsolve` and therefore can't be easily reimplemented into another programing language.

In the next experiment we demonstrate the functionality of our Gröbner basis solution on real-like satellite parameters. We need circular satellite trajectory for a simple example. We have selected International Space Station (ISS) which has trajectory close to circular. For our example, we have specified the circular trajectory movement by mean radius

	Unknowns	Free Parameters	#solutions	G-J elimination
1	$f_{11}, g_{11}, g_{12}, h_{11}, k_{11}$	$h_{21}, k_{12}, k_{21}, k_{22}$	1	5×6
2	$f_{11}, g_{11}, g_{12}, h_{11}, k_{12}$	$h_{21}, k_{11}, k_{21}, k_{22}$	2	10×12
3	$f_{11}, g_{11}, g_{12}, h_{11}, k_{21}$	$h_{21}, k_{11}, k_{12}, k_{22}$	1	5×6
4	$f_{11}, g_{11}, g_{12}, h_{11}, k_{22}$	$h_{21}, k_{11}, k_{12}, k_{21}$	4	20×24
5	$f_{11}, g_{11}, g_{12}, h_{21}, k_{11}$	$h_{11}, k_{12}, k_{21}, k_{22}$	3	20×23
6	$f_{11}, g_{11}, g_{12}, h_{21}, k_{12}$	$h_{11}, k_{11}, k_{21}, k_{22}$	4	20×24
7	$f_{11}, g_{11}, g_{12}, h_{21}, k_{21}$	$h_{11}, k_{11}, k_{12}, k_{22}$	1	5×6
8	$f_{11}, g_{11}, g_{12}, h_{21}, k_{22}$	$h_{11}, k_{11}, k_{12}, k_{21}$	3	10×13
9	$f_{11}, g_{11}, g_{12}, k_{11}, k_{12}$	$h_{11}, h_{21}, k_{21}, k_{22}$	1	5×6
10	$f_{11}, g_{11}, g_{12}, k_{11}, k_{22}$	$h_{11}, h_{21}, k_{12}, k_{21}$	3	5×8
11	$f_{11}, g_{11}, g_{12}, k_{12}, k_{21}$	$h_{11}, h_{21}, k_{11}, k_{22}$	3	5×8
12	$f_{11}, g_{11}, g_{12}, k_{12}, k_{22}$	$h_{11}, h_{21}, k_{11}, k_{21}$	3	5×8
13	$f_{11}, g_{11}, g_{12}, k_{21}, k_{22}$	$h_{11}, h_{21}, k_{11}, k_{12}$	3	5×8
14	$f_{11}, g_{11}, h_{11}, h_{21}, k_{11}$	$g_{12}, k_{12}, k_{21}, k_{22}$	1	5×6
15	$f_{11}, g_{11}, h_{11}, h_{21}, k_{12}$	$g_{12}, k_{11}, k_{21}, k_{22}$	3	20×23
16	$f_{11}, g_{11}, h_{11}, h_{21}, k_{21}$	$g_{12}, k_{11}, k_{12}, k_{22}$	3	10×13
17	$f_{11}, g_{11}, h_{11}, h_{21}, k_{22}$	$g_{12}, k_{11}, k_{12}, k_{21}$	3	20×23
18	$f_{11}, g_{11}, h_{11}, k_{11}, k_{12}$	$g_{12}, h_{21}, k_{21}, k_{22}$	1	5×6
19	$f_{11}, g_{11}, h_{11}, k_{11}, k_{21}$	$g_{12}, h_{21}, k_{12}, k_{22}$	1	5×6
20	$f_{11}, g_{11}, h_{11}, k_{11}, k_{22}$	$g_{12}, h_{21}, k_{12}, k_{21}$	2	5×7
21	$f_{11}, g_{11}, h_{11}, k_{12}, k_{21}$	$g_{12}, h_{21}, k_{11}, k_{22}$	2	10×12
22	$f_{11}, g_{11}, h_{11}, k_{12}, k_{22}$	$g_{12}, h_{21}, k_{11}, k_{21}$	4	15×19
23	$f_{11}, g_{11}, h_{11}, k_{21}, k_{22}$	$g_{12}, h_{21}, k_{11}, k_{12}$	4	15×19
24	$f_{11}, g_{11}, h_{21}, k_{11}, k_{12}$	$g_{12}, h_{11}, k_{21}, k_{22}$	4	15×19
25	$f_{11}, g_{11}, h_{21}, k_{11}, k_{21}$	$g_{12}, h_{11}, k_{12}, k_{22}$	3	15×18
26	$f_{11}, g_{11}, h_{21}, k_{11}, k_{22}$	$g_{12}, h_{11}, k_{12}, k_{21}$	5	20×25
27	$f_{11}, g_{11}, h_{21}, k_{12}, k_{21}$	$g_{12}, h_{11}, k_{11}, k_{22}$	3	20×23
28	$f_{11}, g_{11}, h_{21}, k_{12}, k_{22}$	$g_{12}, h_{11}, k_{11}, k_{21}$	5	20×25
29	$f_{11}, g_{11}, h_{21}, k_{21}, k_{22}$	$g_{12}, h_{11}, k_{11}, k_{12}$	3	15×18
30	$f_{11}, g_{11}, k_{11}, k_{12}, k_{21}$	$g_{12}, h_{11}, h_{21}, k_{22}$	3	15×18
31	$f_{11}, g_{11}, k_{11}, k_{12}, k_{22}$	$g_{12}, h_{11}, h_{21}, k_{21}$	4	5×9
32	$f_{11}, g_{11}, k_{11}, k_{21}, k_{22}$	$g_{12}, h_{11}, h_{21}, k_{12}$	3	15×18
33	$f_{11}, g_{11}, k_{12}, k_{21}, k_{22}$	$g_{12}, h_{11}, h_{21}, k_{11}$	4	5×9
34	$f_{11}, g_{12}, h_{11}, h_{21}, k_{11}$	$g_{11}, k_{12}, k_{21}, k_{22}$	1	5×6
35	$f_{11}, g_{12}, h_{11}, h_{21}, k_{12}$	$g_{11}, k_{11}, k_{21}, k_{22}$	3	10×13
36	$f_{11}, g_{12}, h_{11}, h_{21}, k_{21}$	$g_{11}, k_{11}, k_{12}, k_{22}$	1	5×6
37	$f_{11}, g_{12}, h_{11}, h_{21}, k_{22}$	$g_{11}, k_{11}, k_{12}, k_{21}$	3	10×13
38	$f_{11}, g_{12}, h_{11}, k_{11}, k_{12}$	$g_{11}, h_{21}, k_{21}, k_{22}$	1	5×6
39	$f_{11}, g_{12}, h_{11}, k_{11}, k_{21}$	$g_{11}, h_{21}, k_{12}, k_{22}$	1	5×6
40	$f_{11}, g_{12}, h_{11}, k_{11}, k_{22}$	$g_{11}, h_{21}, k_{12}, k_{21}$	2	5×7
41	$f_{11}, g_{12}, h_{11}, k_{12}, k_{21}$	$g_{11}, h_{21}, k_{11}, k_{22}$	4	20×24
42	$f_{11}, g_{12}, h_{11}, k_{12}, k_{22}$	$g_{11}, h_{21}, k_{11}, k_{21}$	3	18×21
43	$f_{11}, g_{12}, h_{11}, k_{21}, k_{22}$	$g_{11}, h_{21}, k_{11}, k_{12}$	6	15×21
44	$f_{11}, g_{12}, h_{21}, k_{11}, k_{12}$	$g_{11}, h_{11}, k_{21}, k_{22}$	4	14×18
45	$f_{11}, g_{12}, h_{21}, k_{11}, k_{21}$	$g_{11}, h_{11}, k_{12}, k_{22}$	2	32×34
46	$f_{11}, g_{12}, h_{21}, k_{11}, k_{22}$	$g_{11}, h_{11}, k_{12}, k_{21}$	3	20×23
47	$f_{11}, g_{12}, h_{21}, k_{12}, k_{21}$	$g_{11}, h_{11}, k_{11}, k_{22}$	1	5×6
48	$f_{11}, g_{12}, h_{21}, k_{12}, k_{22}$	$g_{11}, h_{11}, k_{11}, k_{21}$	3	15×18
49	$f_{11}, g_{12}, h_{21}, k_{21}, k_{22}$	$g_{11}, h_{11}, k_{11}, k_{12}$	1	5×6
50	$f_{11}, g_{12}, k_{11}, k_{12}, k_{21}$	$g_{11}, h_{11}, h_{21}, k_{22}$	4	5×9
51	$f_{11}, g_{12}, k_{11}, k_{12}, k_{22}$	$g_{11}, h_{11}, h_{21}, k_{21}$	1	5×6
52	$f_{11}, g_{12}, k_{11}, k_{21}, k_{22}$	$g_{11}, h_{11}, h_{21}, k_{12}$	3	15×18
53	$f_{11}, g_{12}, k_{12}, k_{21}, k_{22}$	$g_{11}, h_{11}, h_{21}, k_{11}$	4	5×9
54	$f_{11}, h_{11}, h_{21}, k_{11}, k_{12}$	$g_{11}, g_{12}, k_{21}, k_{22}$	1	5×6
55	$f_{11}, h_{11}, h_{21}, k_{11}, k_{21}$	$g_{11}, g_{12}, k_{12}, k_{22}$	2	5×7
56	$f_{11}, h_{11}, h_{21}, k_{11}, k_{22}$	$g_{11}, g_{12}, k_{12}, k_{21}$	2	5×7
57	$f_{11}, h_{11}, h_{21}, k_{12}, k_{21}$	$g_{11}, g_{12}, k_{11}, k_{22}$	2	15×17
58	$f_{11}, h_{11}, h_{21}, k_{12}, k_{22}$	$g_{11}, g_{12}, k_{11}, k_{21}$	3	5×8
59	$f_{11}, h_{11}, h_{21}, k_{21}, k_{22}$	$g_{11}, g_{12}, k_{11}, k_{12}$	1	5×6
60	$f_{11}, h_{11}, k_{11}, k_{12}, k_{21}$	$g_{11}, g_{12}, h_{21}, k_{22}$	2	5×7
61	$f_{11}, h_{11}, k_{11}, k_{12}, k_{22}$	$g_{11}, g_{12}, h_{21}, k_{21}$	2	5×7
62	$f_{11}, h_{11}, k_{11}, k_{21}, k_{22}$	$g_{11}, g_{12}, h_{21}, k_{12}$	3	5×8
63	$f_{11}, h_{11}, k_{12}, k_{21}, k_{22}$	$g_{11}, g_{12}, h_{21}, k_{11}$	4	5×9
64	$f_{11}, h_{21}, k_{11}, k_{12}, k_{21}$	$g_{11}, g_{12}, h_{11}, k_{22}$	4	5×9
65	$f_{11}, h_{21}, k_{11}, k_{12}, k_{22}$	$g_{11}, g_{12}, h_{11}, k_{21}$	4	5×9
66	$f_{11}, h_{21}, k_{11}, k_{21}, k_{22}$	$g_{11}, g_{12}, h_{11}, k_{12}$	4	5×9
67	$f_{11}, h_{21}, k_{12}, k_{21}, k_{22}$	$g_{11}, g_{12}, h_{11}, k_{11}$	2	10×12
68	$f_{11}, k_{11}, k_{12}, k_{21}, k_{22}$	$g_{11}, g_{12}, h_{11}, h_{21}$	5	5×10

TABLE I
DETAILED DESCRIPTION OF DIFFERENT SOLVERS.

```
% Satellite trajectory control by pole placement

cfg = gbs_InitConfig();

% known variables
syms l1 l2 l3 l4 l5

syms v r0 o0

% control variables
syms f11 g11 g12 h11 h21 k11 k12 k21 k22

l = [l1 l2 l3 l4 l5]

% build equations
for i=1:5
    eq(i) =  l(i)^5*v*r0 + l(i)^3*v*r0*o0^2 - l(i)^4*v*r0*f11 - ...
       l(i)^2*v*r0*o0^2*f11 - 2*l(i)*o0*h11*g11 + l(i)^2*h21*g11 - ...
       3*o0^2*h21*g11 - 2*l(i)^2*o0*h11*g12 + l(i)^3*h21*g12 - ...
       3*l(i)*o0^2*h21*g12 - 2*l(i)^2*o0*k11 + 2*l(i)*o0*f11*k11 - ...
       2*l(i)^3*o0*k12 + 2*l(i)^2*o0*f11*k12 + l(i)^3*k21 - ...
       3*l(i)*o0^2*k21 - l(i)^2*f11*k21 + 3*o0^2*f11*k21 + ...
       l(i)^4*k22 - 3*l(i)^2*o0^2*k22 - l(i)^3*f11*k22 + 3*l(i)*o0^2*f11*k22;
end


groups = [];
n_sols = [];

must_unknown = {'f11'};
known = {'l1' 'l2' 'l3' 'l4' 'l5' 'v' 'r0' 'o0'};
allUnknown = {'g11' 'g12' 'h11' 'h21' 'k11' 'k12' 'k21' 'k22'};
unkChoices = nchoosek(1:length(allUnknown), 4)';

solIdx = 1;
for unk = unkChoices

    current_unknown = [must_unknown allUnknown(unk)];

    current_unknown

    unk_complement = setdiff(allUnknown, current_unknown);
    current_known = [known unk_complement];
    [res export n_sol(solIdx)] = gbs_CreateCode(['satellite_pole' ...
        int2str(solIdx) ], eq, current_known, current_unknown, groups, cfg );
    solIdx = solIdx + 1;
end
```

Fig. 3. Input Matlab script for the automatic generator of Gröbner basis solvers

and mean motion of ISS. Parameters of the circular motion are radius of the circle $r_0 = 6794209$ km, angular velocity $\omega_0 = 0.001127$ rad/s and mass of the satellite $v = 450000$ kg.

State space representation of the satellite dynamic system is numerically unsuitable when we use ISS parameters in SI units. Therefore we have used the transformation of time from seconds to hours [h] and distance units from meters to thousands of kilometers [kkm]. Transformed parameters of the satellite motion are radius of the circle $r_0 = 6.794209$ kkm and angular velocity $\omega_0 = 4.058687$ rad/h. The parameter v combines together the mass of the satellite and input force gain. We have defined unit input of the system as 100kN. The parameter v is 4.5 hundreds of tons. We have created the state-space model of the satellite dynamic with the parameters based on equations (9), (10) and (11).

The system performance is documented by response to pulse of input error force in radial direction. The value of the force has been 500N and the pulse duration has been 60 second. The force impulse has been applied at the time equal to 0.2 hour (720 s). The response of the system without feedback is shown on Figure 4.

For the created dynamic model, we have designed dynamic output feedback by pole placement method as described

Solver	f_{11}	g_{11}	h_{11}	m_{11}	m_{22}
Dixon resultant [12]	-7.80999	-473.11582	-11.05439	-719.88274	8.26297
	-8.97886	-503.50636	-12.04822	-719.31480	7.18729
Gröbner basis	-7.80999	-473.11582	-11.05439	-719.88274	8.26297
	-8.97886	-503.50636	-12.04822	-719.31480	7.18729

TABLE II
RESULTS OF THE DIXON RESULTANT SOLVER AND THE PROPOSED GRÖBNER BASIS SOLVER.

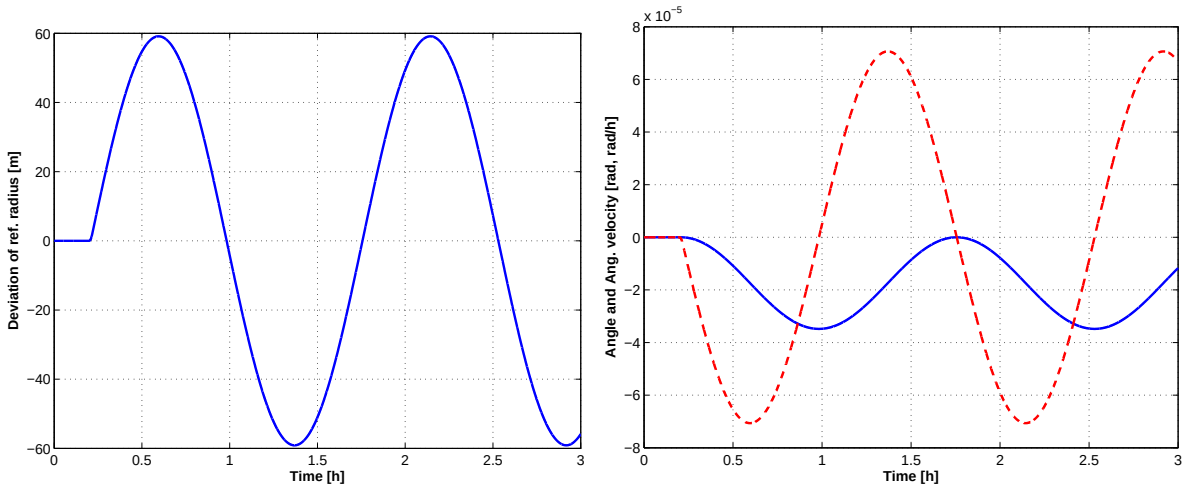

Fig. 4. Response of satellite dynamic system to radial force pulse (60s, 500N): (Left) Orbit radial distortion and (Right) distortion of position angle (blue line) and angular velocity (red dashed line).

in Section 2. We suggest stable poles with one complex conjugate pole pair:

$$\lambda_1 = -7 + 4i,\ \lambda_2 = -7 - 4i,\ \lambda_3 = -5,\ \lambda_4 = -7,\ \lambda_5 = -3. \tag{18}$$

The dynamic compensator (12) is underdetermined, i.e. it results in the underdetermined system of polynomial equations (14). Therefore we have to select free parameters and set their values.

In this experiment we have tested two of our Gröbner basis solvers. The first one is the solver 20 from Table I, i.e. the same solver as in the previous experiment. In this solver we have set

$$g_{12} = 1,\ h_{21} = 1,\ k_{12} = 1,\ k_{21} = 290. \tag{19}$$

The second one is the solver 19 from Table I. This solver returns only one solution and therefore ensures real solution for reasonable inputs. In this solver we have set

$$g_{12} = 1,\ h_{21} = 1,\ k_{12} = 1,\ k_{22} = 70. \tag{20}$$

For the solver 19 we have tested solutions for different values of the parameter k_{22} (from -100 to 100). As expected, we have obtained real solutions in all these cases. We have set the value of k_{22} to 70 because this value offers a solution which minimizes the maximal absolute values of parameters.

We used the two tested solvers with the corresponding values of free parameters to obtain solutions to unknown parameters of the dynamic compensator (12).

For the solver 20 we have obtained two solutions. The first solution

$$\mathtt{F} = \begin{bmatrix} -15.2963 \end{bmatrix},\ \mathtt{G} = \begin{bmatrix} -4761.7 & 1 \end{bmatrix},\ \mathtt{H} = \begin{bmatrix} -1.1320 \\ 1.0000 \end{bmatrix},\ \mathtt{K} = \begin{bmatrix} -780.6398 & 1 \\ 290 & 32.3283 \end{bmatrix}, \tag{21}$$

and the second solution

$$\mathtt{F} = \begin{bmatrix} -13.7037 \end{bmatrix},\ \mathtt{G} = \begin{bmatrix} -4299.9 & 1 \end{bmatrix},\ \mathtt{H} = \begin{bmatrix} -1.0568 \\ 1.0000 \end{bmatrix},\ \mathtt{K} = \begin{bmatrix} -809.6220 & 1 \\ 290 & 36.0856 \end{bmatrix}. \tag{22}$$

For the solver 19 we have obtained only one solution

$$\mathtt{F} = \begin{bmatrix} 0.6723 \end{bmatrix},\ \mathtt{G} = \begin{bmatrix} 233.2827 & 1 \end{bmatrix},\ \mathtt{H} = \begin{bmatrix} -31.0790 \\ 1.0000 \end{bmatrix},\ \mathtt{K} = \begin{bmatrix} -1040.5195 & 1 \\ 831.5678 & 70 \end{bmatrix}. \tag{23}$$

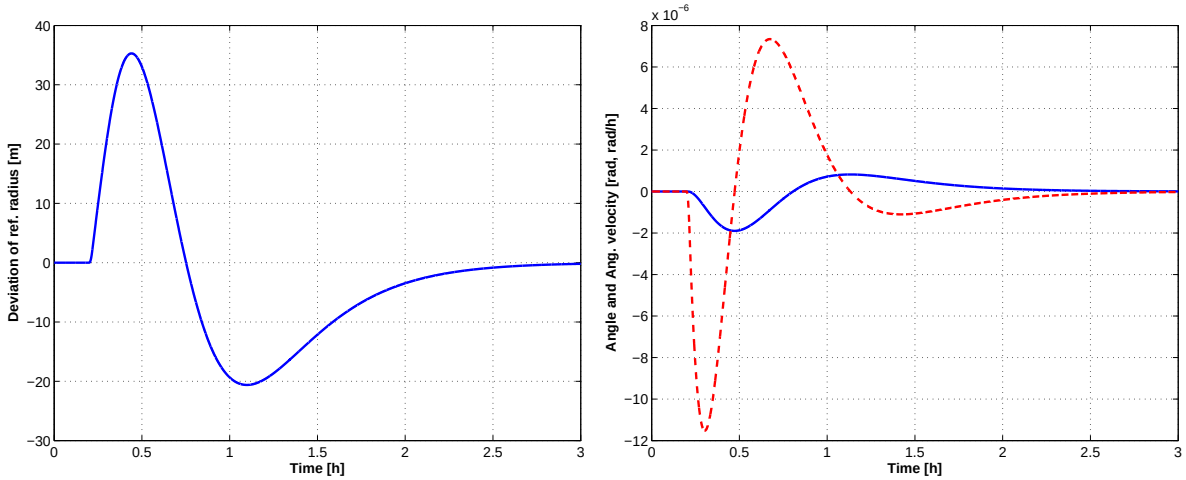

Fig. 5. Response of dynamic system with compensator designed by the solver 20 to radial force pulse (60s, 500N): (Left) Orbit radial distortion and (Right) distortion of position angle (blue line) and angular velocity (red dashed line).

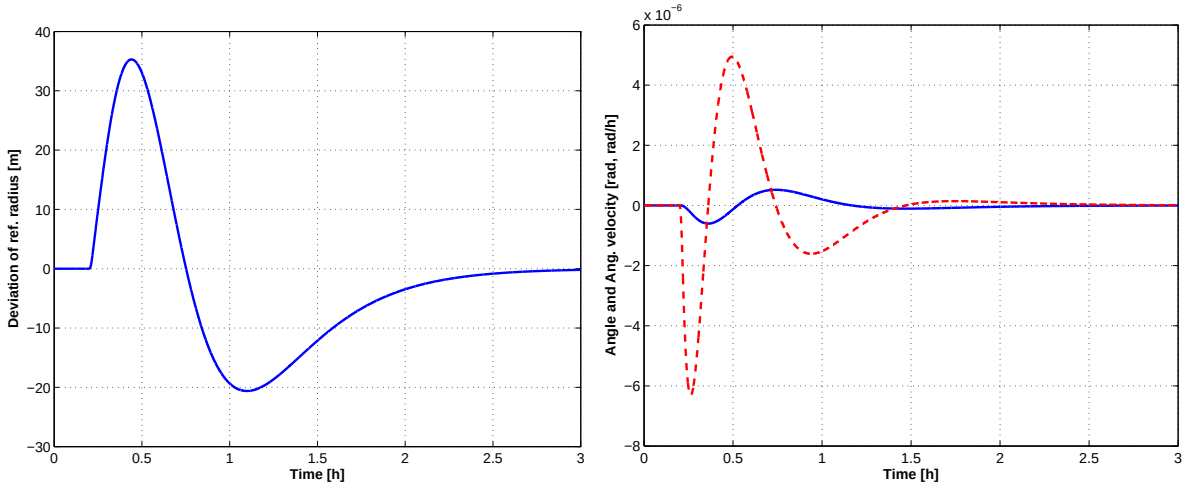

Fig. 6. Response of dynamic system with compensator designed by the solver 19 to radial force pulse (60s, 500N): (Left) Orbit radial distortion and (Right) distortion of position angle (blue line) and angular velocity (red dashed line).

The response of the system with dynamic compensator (21) and closed feedback obtained from the first solution returned by the solver 20 is shown on Figure 5. We can see, that after error disturbance, the distortion of radius Figure 5 (Left), distortion of position angle and angular speed system Figure 5 (Right) have nonzero value. Their values are returning to zero with suggested dynamic.

The response of the system with dynamic compensator (21) and closed feedback obtained using the solver 19 is shown on Figure 6. We can see, that the characteristics corresponds to the previous example in Figure 5 due to the same required system poles.

7. Conclusion

In this work, we have presented Gröbner basis solutions to the important problem of computation of a dynamic compensator for the satellite.

Since the problem results in an underconstrained system of polynomial equations wh have proposed solutions for different combinations of free input parameters. We have shown that the Gröbner basis method for solving systems of polynomial equations leads to simple solutions for all combinations of free parameters. These solutions require to perform only the Gauss-Jordan elimination of a small matrix and computation of roots of a single variable polynomial. The maximum degree of this polynomial is not greater than six in general but for most combinations of the input free parameters its degree is even lower. Our approach is particularly useful for testing different parameters of the

compensator and for developing numerically stable compensators, when, e.g., state space needs to be transformed to a different coordinate system.

ACKNOWLEDGMENT

This work was supported by the Missile Defense Agency (USA), contract 201-0528-01, by the project PRoViDE EU FP7-SPACE-312377 and by Grant Agency of the CTU Prague project SGS12/191/OHK3/3T/13.

REFERENCES

[1] J. Brasch, F. and J. Pearson. Pole placement using dynamic compensators. *Automatic Control, IEEE Transactions on*, 15(1):34–43, 1970.

[2] M. Bujnak, Z. Kukelova, and T. Pajdla. Making Minimal Solvers Fast. In *CVPR'12*, 2012.

[3] C. Byrnes. Pole assignment by output feedback. In H. Nijmeijer and J. Schumacher, editors, *Three Decades of Mathematical System Theory*, volume 135 of *Lecture Notes in Control and Information Sciences*, pages 31–78. Springer Berlin Heidelberg, 1989.

[4] M. Byröd, K. Josephson, and K. Åström. A column-pivoting based strategy for monomial ordering in numerical gröbner basis calculations. In *ECCV'08*, 2008.

[5] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer, 2nd edition, 2005.

[6] B. Datta. *Numerical Methods for Linear Control Systems*. Elsevier Science, 2004.

[7] R. Dorf and R. Bishop. *Modern Control Systems*. Pearson Prentice Hall, 2008.

[8] T. Kailath. *Linear Systems*. Prentice-Hall information and system sciences series. Prentice Hall International, 1998.

[9] K.Neokleous. Modeling and control of a satellites geostationary orbit. Master's thesis, Lulea University of Technology, Czech Technical University in Prague, 2007.

[10] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. In *ECCV'08, Part III*, volume 5304 of *Lecture Notes in Computer Science*, 2008.

[11] Z. Kukelova and T. Pajdla. A minimal solution to radial distortion autocalibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2410–2422, December 2011.

[12] B. Paláncz. Application of dixon resultant to satellite trajectory control by pole placement. *J. Symb. Comput.*, 50:79–99, Mar. 2013.

[13] B. Paláncz. Numeric-symbolic solution for satellite trajectory control by pole placement. *Periodica Polytechnica Civil Engineering*, 57(1):21–26, 2013.

[14] J. Rosenthal and X. A. Wang. Output feedback pole placement with dynamic compensators. *IEEE Trans. Automat. Contr.*, 41:830–843, 1995.

[15] J. Verschelde. Algorithm 795: Phcpack: a general-purpose solver for polynomial systems by homotopy continuation. *ACM Trans. Math. Softw.*, 25(2):251–276, June 1999.

[16] J. Verschelde and Y. Wang. Numerical homotopy algorithms for satellite trajectory control by pole placement. In *In Proceedings of MTNS 2002, Mathematical Theory of Networks and Systems, 2002. Notre Dame*, pages 12–16, 2002.

[17] J. Verschelde and Y. Wang. Computing dynamic output feedback laws. *IEEE Trans. Automat. Contr.*, pages 1393–1397, 2004.