Numerical Homotopy Algorithms for Satellite Trajectory Control by Pole Placement[1]

Jan Verschelde[2] and Yusong Wang[3]
Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago
851 South Morgan (M/C 249)
Chicago, IL 60607-7045, USA

Abstract

The aim of this paper is to illustrate the application of numerical homotopy algorithms to control the trajectory of a satellite. We design output feedback laws via pole placement. The output feedback laws are computed by numerical homotopy algorithms, which are based on enumerative geometry. We combine the use of MATLAB with PHCpack.

2000 Mathematics Subject Classification. Primary 93B55. Secondary 14Q99, 65H10, 68W30, 93B27.

Key words and phrases. numerical homotopy algorithms, pole placement, satellite trajectory control.

1 Introduction

The satellite trajectory control problem is a frequently occurring illustration in textbooks [4, 8]. For the purposes of this paper, the textbook models suffice, see [1] for a more realistic model.

In section two we regard the output feedback laws for this problem as lines meeting four given lines in projective 3-space. In the third section we outline the numerical homotopy algorithms we use to find the feedback laws. In the last two sections we show how we combine MATLAB and PHCpack [15] to find the feedback laws and to study their robustness.

While the scale of the control problem is rather modest, this paper is the first – to the best of our knowledge – to apply the new Pieri homotopies (see [6], [7] and [10]) to a control engineering problem, which appeared as case study in the literature.

Our methodology extends to the design of controllers with internal states, as our software also implements the dynamic pole placement problem [13].

[1]The authors gratefully acknowledge the support of this work by the Campus Research Board of UIC. This material is based upon work supported by the National Science Foundation under Grant No. 0105739 and Grant No. 0134611.

[2]E-mail: jan@math.uic.edu, URL: http://www.math.uic.edu/∼jan

[3]E-mail: ywang25@uic.edu

2 A Geometric View on the Pole Placement Problem

The relevance of algebraic geometry for the pole placement problem was realized in the seventies, see the survey [2]. In this section we state the problem and derive the format of the equations we solve to get the feedback laws.

We assume we are given a linear system with m inputs $\mathbf{u} \in \mathbb{R}^m$, and p outputs $\mathbf{y} \in \mathbb{R}^p$ by three matrices: $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, and $C \in \mathbb{R}^{p \times n}$, where n equals the number of internal states stored by the vector $\mathbf{x} \in \mathbb{R}^n$. These three matrices define the system of linear first order differential equations : $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$, with output $\mathbf{y} = C\mathbf{x}$. We assume the system is completely observable and controllable. The most simple output feedback law can be denoted by a single matrix $F \in \mathbb{C}^{m \times p}$, which leads to the feedback $\mathbf{u} = F\mathbf{y}$. Elimination of $\mathbf{u}$ and $\mathbf{y}$ leads to the following *closed-loop system*:

$$\dot{\mathbf{x}}(t) = (A + BFC)\mathbf{x}(t), \quad \dot{\mathbf{x}}(t) = \frac{d}{dt}\mathbf{x}(t), \tag{2.1}$$

whose behavior is determined by the eigenvalues of $A + BFC$.

Our problem is then to find feedback laws F, for given eigenvalues $\lambda_i = 1, 2, \ldots, n$. Thus F is defined by the equations

$$\det(\lambda_i I_n - (A + BFC)) = 0, \quad i = 1, 2, \ldots, n, \tag{2.2}$$

where I_n is the identity matrix of size n. To derive the geometric model, we consider equations equivalent to (2.2):

$$\det \begin{pmatrix} \lambda_i I_n - (A + BFC) & BF & -B \\ 0 & I_p & 0 \\ 0 & 0 & I_m \end{pmatrix} = 0, \quad i = 1, 2, \ldots, n. \tag{2.3}$$

The matrices BF and $-B$ will serve us well in the reduction of $\lambda I_n - (A + BFC)$ to I_n. First we eliminate the BFC: we right multiply the second column of the matrix in (2.3) by C and add the result to the first column. We remove BF by right multiplication of the third column by F and addition of the result to the second column. So, we obtain

$$\det \begin{pmatrix} \lambda_i I_n - A & 0 & -B \\ C & I_p & 0 \\ 0 & F & I_m \end{pmatrix} = 0, \quad i = 1, 2, \ldots, n. \tag{2.4}$$

We multiply the first row of the matrix in (2.4) by $(\lambda_i I_n - A)^{-1}$ (inverse exists if λ_i is not an eigenvalue of A) and then we eliminate C, subtracting C times the first from the second row. Thus,

$$\det \begin{pmatrix} I_n & 0 & -(\lambda_i I_n - A)^{-1}B \\ 0 & I_p & C(\lambda_i I_n - A)^{-1}B \\ 0 & F & I_m \end{pmatrix} = 0, \quad i = 1, 2, \ldots, n, \tag{2.5}$$

or equivalently,

$$\det \begin{pmatrix} I_p & C(\lambda_i I_n - A)^{-1}B \\ F & I_m \end{pmatrix} = 0, \quad i = 1, 2, \ldots, n. \tag{2.6}$$

In this equivalent formulation for our problem the given data: (A, B, C) and λ_i, $i = 1, 2, \ldots, n$ appear separate from the unknown feedback laws F.

For each eigenvalue λ_i, we create a matrix $[C(\lambda_i I_n - A)^{-1}B \quad I_m]^T$ with $m + p$ rows and m columns. We interpret this matrix as the matrix whose columns contain the generators of an m-plane. Then the feedback laws correspond to p-planes in $\mathbb{C}^{m+p}$, and the equations in (2.6) require the p-planes to intersect the given m-planes. For a complete intersection problem: $n = mp$, because we have mp unknown coefficients in $F \in \mathbb{C}^{m \times p}$.

In Figure 1 we show an example in projective three space, where $m = 2$, and $p = 2$. A 2-plane in projective space, spanned by two generators, is a line in affine space. For visualization purposes, the positive orthant of projective real space is mapped into a tetrahedron. Figure 1 is computed with Maple, the worksheet is available via the first author's homepage.

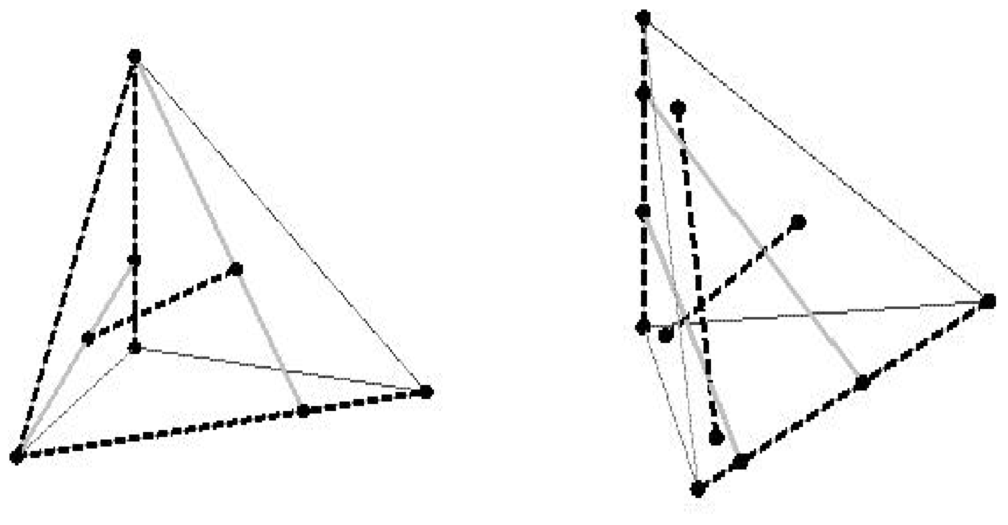

Figure 1: Two (solid) lines meet four given (dashed) lines in three space. At the left, one of the four given lines is in special position, i.e.: one given line touches two other given lines. The configuration at the right is generic. Its solution lines by homotopy continuation. As the special line at the left moves to general position, we continuously adjust the two solution lines so that they keep meeting the four given lines.

3 Numerical Homotopy Algorithms

The development of numerically stable algorithms for the pole placement is stated as an open problem in [14]. Citing [3]: "*the output feedback pole assignment problem (OPAP) is essentially unsolved*".

The use of homotopies to numerically solve the polynomial systems defined by the pole placement was already mentioned in [2]. In this section we outline the three different stages of our method, using the following homotopies:

1. Pieri homotopies to solve one generic instance in complex space.

Geometric problems are often solved by bringing the input configuration from a general to a special position where the solution can be determined by inspection, or by induction to a lower dimensional problem. If the solution paths defined by this deformation stay finite and free of singularities – except possibly for some algebraic set – then the principle of "conservation of number" applies. This principle is used in enumerative geometry to *count* the number of solutions, but leads very naturally to an efficient numerical homotopy algorithm to *compute* all solutions.

For our problem in (2.6), so-called Pieri homotopies were first defined in [6], and then implemented and generalized in [7]. Recently, a more efficient version is described in [10].

2. Cheater's homotopy to solve one specific real instance.

With "real" we do not mean that the numbers need to be real, because eigenvalues with nonzero imaginary parts lead to complex input planes, but that the input data have a physical meaning. For the cheater's homotopy to work, we assume we have already solved a problem with similar structure, but with different coefficients. As it is in general not obvious how to obtain the solutions to such a similar problem, there is some cheating going on. For our problem, Pieri homotopies deliver the solutions to a generic problem instance.

We can define the transition from the generic to the specific as a convex linear combination between two systems of equations, defined in (2.6). In a nonlinear cheater's homotopy, we put the continuation parameter inside the determinant. In both cases, singularities can only occur at the end of the paths.

A more general version of cheater's homotopy [9] is coefficient-parameter polynomial continuation [11], which can also be applied for the next type of homotopies.

3. Natural parameter homotopies for design and sensitivity analysis.

The natural parameters are the eigenvalues of the closed-loop system. As we wish to move those eigenvalues, the corresponding feedback laws are varying continuously as well. As with many design problems, the outcome of the design process can only be evaluated through simulation. For fine tuning and sensitivity analysis of the feedback laws it is important that we do not apply the Pieri homotopies or the Cheater's homotopy from a random complex instance to a real case. Note that eigenvalues are generally chosen to minimize the sensitivity of the closed-loop system to errors [12].

These homotopy methods have been implemented in PHCpack [15]. In the computational experiments MATLAB scripts were used to generate the input files with the matrices for PHCpack. The solutions computed with homotopies were read from file by MATLAB scripts.

4 Solving Equations to Find the Feedback Laws

As in [8], polar coordinates are used for the satellite being in a circular equatorial orbit. The goal of the feedback is to keep the satellite in the same orbit when disturbances such as aerodynamic drag [4] cause it to deviate.

The state vector is $\mathbf{x} = [r \; \dot{r} \; \theta \; \dot{\theta}]$, and the input is $\mathbf{u} = [u_r \; u_t]$, with u_r and u_t respectively the radial and tangential thrusters. The linearized state-space equations are defined by the matrices

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 3\omega_0^2 & 0 & 0 & 2\omega_0 r_0 \\ 0 & 0 & 0 & 1 \\ 0 & -2\omega_0/r_0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 0 & 0 \\ 1/m & 0 \\ 0 & 0 \\ 0 & 1/mr_0 \end{pmatrix}, \tag{4.7}$$

where the radius r_0 and angular velocity ω_0 are such that $r_0^3\omega_0^2$ equals a constant. The mass of the satellite is m. As we now have a system with two inputs, we also like to have two outputs. We can define $C \in \mathbb{R}^{2\times 4}$ as some random matrix – which can be interpreted as a random projection of the states onto a plane. Another choice of C that allows us to find feedback laws is

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.8}$$

This special choice of C is in agreement with the demonstration in [4, pages 659–660] that the satellite is completely controllable with the tangential thruster u_t only, thus without u_r. While this choice "works", we obtained better numerical results with random choices for C.

Values for ω_0 and r_0 are chosen at random. While this may seem unrealistic, our choice can be justified by an appropriate selection of units to measure distances and angles. We choose eigenvalues for the closed-loop systems, two are complex conjugated, and two are real. In particular, our selection of eigenvalues is $(-2+i, -2-i, -5, -7)$. Dividing the eigenvalues with nonzero imaginary part by their length leads to better numerical results. In both cases, the two feedback laws are real.

We use the convenience of the matrix manipulation capabilities in MATLAB to define the input matrices $[C(\lambda_i I_4 - A)^{-1})B \;\; I_2]^T$ for the four chosen eigenvalues λ_i. In (2.6) we multiply the matrix with a random orthogonal matrix to simulate a random coordinate change. In this way, the paths in the homotopies will not diverge because of the special shape of the matrix, characterized by the positioning of the identity matrices.

The results of the homotopy methods for this small 2-input 2-output system come almost immediate. Most of our efforts with an application of this small scale are spent on format conversions. The sequence of operations goes as follows :

```
1. MATLAB generates input planes and writes to a file;
2. the file produced by MATLAB is converted into PHCpack format;
3. with PHCpack output planes are computed and written to a file;
4. the file produced by PHCpack is converted into MATLAB format;
5. the feedback laws are read into MATLAB for processing (see section 5).
```

The first task is done with a MATLAB script (see the appendix). For stages two and four we wrote little C programs.

The computation of the two output feedback laws takes less than half a second on a SUN Ultra-4 Workstation.

For larger systems with more states we will have more feedback laws to compute. For example, with three inputs and three outputs, there are 42 feedback laws, and we can assign 9 eigenvalues. In the Pieri homotopies we trace 148 paths (in 52s 860ms cpu user time) to solve a generic complex instance of the geometric problem. To solve a random real instance, we start the cheater's homotopy at the complex solutions of the generic instance. Tracing those 42 paths takes 1m 52s 120ms cpu user time. The total cpu user time of the two stages is 2m46s840ms. So our approach extends to larger multi-input multi-output systems.

5 Verification and Robustness Study

The first step after the computation of the feedback laws consists in verifying whether the closed-loop system, defined by $\dot{\mathbf{x}}(t) = (A + BFC)\mathbf{x}(t)$ has the desired behavior. This verification consists in comparing the initial given eigenvalues in the pole placement problem with the computed eigenvalues of the matrix $A + BFC$.

For a choice $(-2 + i, -2 - i, -5, -7)$ of the four eigenvalues, the computed eigenvalues for the closed-loop system differ by as much as 10^{-9}. If we choose the eigenvalues as $(\frac{-2+i}{\sqrt{5}}, \frac{-2-i}{\sqrt{5}}, -5, -7)$, the difference between given and computed eigenvalues lowers to 10^{-10}.

It is quite plausible that by cranking up the number of decimal places in the computation, we can wipe out the roundoff errors and bring the computed eigenvalues arbitrarily close to the given ones. However, the corresponding feedback laws may still be useless for practical purposes as the matrices A and B which define the model are usually subject to perturbations. And even if we knew the parameters ω_0 and r_0 exactly, we have to keep in mind that our model is only an approximate linearized model. To investigate the quality of the computed feedback laws, in a second stage, we compute condition numbers.

To estimate the condition numbers measuring the sensitivity of the feedback laws with respect to changes in the parameters m, ω, and r, we can look at the condition numbers of the solutions of the polynomial system. These numbers range between 10^6 and 10^7. Note that these estimates ignore the structure of the problem and are therefore not very accurate. Moreover, we are interested in the sensitivity of the eigenvalues of the resulting closed-loop loop system.

In our robustness study, we follow the approach taken in [12]. If we perturb A and B adding respectively ΔA and ΔB, then the error on the eigenvalues λ_i can be estimated by

$$\frac{||\Delta A + (\Delta B)F||}{|\mathbf{y}_i^H \mathbf{x}_i|}, \quad i = 1, 2, \ldots, n, \tag{5.9}$$

where vectors $\mathbf{x}_i$ and $\mathbf{y}_i$ denote the unit right and left eigenvectors of the closed-loop system $A+BFC$. See [5, page 323] for the derivation of $|\mathbf{y}_i^H \mathbf{x}_i|$ as the reciprocal of the condition number for the eigenvalue λ_i.

The perturbation matrices ΔA and ΔB respect the specific problem structure, i.e.: they are determined by Δm, $\Delta\omega$ and Δr. In ΔA and ΔB, we work with relative errors:

$$\Delta A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 6\omega_0^2 \left|\frac{\Delta\omega}{\omega_0}\right| & 0 & 0 & 2\omega_0 r_0 \left(\left|\frac{\Delta\omega}{\omega_0}\right| + \left|\frac{\Delta r}{r_0}\right|\right) \\ 0 & 0 & 0 & 1 \\ 0 & \frac{2\omega_0}{r_0}\left(\left|\frac{\Delta\omega}{\omega_0}\right| + \left|\frac{\Delta r}{r_0}\right|\right) & 0 & 0 \end{pmatrix} \tag{5.10}$$

and

$$\Delta B = \begin{pmatrix} 0 & 0 \\ \frac{1}{m}\left|\frac{\Delta m}{m}\right| & 0 \\ 0 & 0 \\ 0 & \frac{1}{mr_0}\left(\left|\frac{\Delta m}{m}\right| + \left|\frac{\Delta r}{r_0}\right|\right) \end{pmatrix}. \tag{5.11}$$

If $\left|\frac{\Delta m}{m}\right|$, $\left|\frac{\Delta\omega}{m}\right|$, and $\left|\frac{\Delta r}{r_0}\right|$ are all equal to 10^{-5}, then the error on one real eigenvalue is less than 10%. For 10^{-6}, the errors on the two real eigenvalues are less than 10%. The errors on all eigenvalues are less than 10% for magnitudes 10^{-7} or lower.

See [16] for other techniques to determine the range of the size of tolerable perturbations.

References

[1] R.H. Bishop, S.J. Paynter, and J.W. Sunkel. Adaptive control of space station with control moment gyros. *IEEE Control Systems*, pages 23–27, October 1992.

[2] C.I. Byrnes. Pole assignment by output feedback. In *Three Decades of Mathematical Systems Theory*, edited by H. Nijmacher and J.M. Schumacher, pages 13–78. Springer–Verlag, Berlin, 1989.

[3] E.K. Chu. Optimization and pole assignment in control system design. *International Journal of Applied Mathematics and Computer Science* 11(5):1035–1053, 2001.

[4] D.C. Dorf and R.H. Bishop. *Modern Control Systems.* Addison-Wesley, 1998.

[5] G.H. Golub and C.F. Van Loan. *Matrix Computations.* Third Edition. The Johns Hopkins University Press, 1996.

[6] B. Huber, F. Sottile, and B. Sturmfels. Numerical Schubert calculus. *J. of Symbolic Computation* 26(6):767–788, 1998.

[7] B. Huber and J. Verschelde. Pieri homotopies for problems in enumerative geometry applied to pole placement in linear systems control. *SIAM J. Control Optim.* 38(4):1265–1287, 2000.

[8] T. Kailath. *Linear Systems.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1980.

[9] T.Y. Li, T. Sauer, and J.A. Yorke. The cheater's homotopy: an efficient procedure for solving systems of polynomial equations. *SIAM J. Numer. Anal.* 26(5):1241–1251, 1989.

[10] T.Y. Li, X. Wang, and M. Wu. Numerical Schubert calculus by the Pieri homotopy algorithm. To appear in *SIAM J. Numer. Anal.*

[11] A.P. Morgan and A.J. Sommese. Coefficient-parameter polynomial continuation. *Appl. Math. Comput.*, 29(2):123–160, 1989. Errata: *Appl. Math. Comput.* 51:207(1992).

[12] W.F. Moss. Vertical stabilization of a rocket on a movable platform. In *Applied Mathematical Modeling. A multidisciplinary approach*, edited by D.R. Shier and K.T. Wallenius, pages 363–381. Chapman & Hall/CRC 2000.

[13] M.S. Ravi, J. Rosenthal, and X. Wang. Dynamic pole placement assignment and Schubert calculus. *SIAM J. Control Optim.* 34(3):813–832, 1996.

[14] J. Rosenthal and J.C. Willems. Open problems in the area of pole placement. In *Open Problems in Mathematical Systems and Control Theory*, edited by V.D. Blondel, E.D. Sontag, M. Vidyasagar, and J.C. Willems, pages 181–191. Springer–Verlag, 1998.

[15] J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Transactions on Mathematical Software* 25(2):251–276, 1999. Software available at `http://www.math.uic.edu/~jan`.

[16] R.K. Yedavalli. Robust control design for aerospace applications. *IEEE Transactions on Aerospace and Electronic Systems* 25(3):314–324, 1989.

Appendix: Matlab Script to generate the input planes

```
% Communications Satellite in a circular equatorial orbit
% This MATLAB script generates the input planes for the Pieri
% homotopy algorithm and writes those to the file Result.txt.
% The choice of eigenvalues is in the variable lambda below.

delete Result.txt;
diary Result.txt;
diary on;
n = 4;                 % dimension of ambient space
m = 2;                 % dimension of input space
p = 2;                 % dimension of output space
w = 0.345354;          % angular velocity
r = 1.2342;            % radius is distance of satellite to center of earth
ms = 0.74564;          % mass of satellite
A=[0      1       0  0
   3*w^2  0       0  2*w*r
   0      0       0  1
   0      -2*w/r  0  0];
B=[0     0
   1/ms  0
   0     0
   0     1/(ms*r)];
C = rand(2,4);
format long
lambda = [complex(-2,1)  complex(-2,-1)  -5  -7];
ct = orth(rand(4));  % random coordinate change
Id = eye(m);
for j=1:n
    M = lambda(j)*eye(n)-A;
    Result = C/M*B;
    for i=1:m
        Result(i+p,:) = Id(i,:);
    end;
    disp(ct*Result);
end;
diary off;
```

Note that the random coordinate change (realized by the random matrix `ct` in the script) is needed to make sure the solution will fit the localization pattern of the Pieri homotopies.