APPENDIX A

Numerical aspects in geometric control

A.1. Introduction

The power of geometric control theory is due to its direct treatment of the fundamental structural questions at the root of many important control synthesis problems. The level of abstraction is high enough to separate structural questions from computational ones, thereby achieving remarkable results. It allows then, first to understand the nature of control problems and then, with the help of computational tools solve the different problems encountered. Rigorous and extensive descriptions of numerical algorithms in general can be found in [**8**].

As the reader has probably noticed, the main results in geometric control theory are always expressed in terms of maximal (A, B) invariant and reachability subspaces, contained in the kernel of some other transformation. Take for example the $\mathcal{V}^*$ algorithm explained in Theorem 3.3 and concretized into matrix computations form. Notice that the main step of the algorithm is the computation of $\mathcal{V}_{i+1} = \ker C \cap A^{-1}(\text{Im } B + \mathcal{V}_i)$. Thus any implementation of the algorithm needs to solve, at least, the following problems:

(1) find a basis for the span of the column vectors of two matrices,
(2) compute a basis of the intersection of two subspaces,
(3) compute a basis of the preimage of a subspace under a linear map.

At the heart of the above problems lies the difficulty in finding the rank of a given matrix. In fact, this is meaningless if we are dealing with numbers of limited precision, as are the floating point representation of numbers in computers, and finite precision arithmetics. Instead we should be able to compute the *effective rank* of a given matrix, which is the rank of a matrix which within the precision of the data and computations *could be* the given matrix.

To get numerically accurate answers from a computer you typically need

(1) a numerically stable algorithm in finite-precision arithmetic,
(2) a well conditioned problem,
(3) a good software implementation of the algorithm.

Among the well-proven and available libraries of computer routines we mention EISPACK, LINPACK, LAPACK, SLATEC, and many others. The commercial product MATLAB® uses many routines of the first mentioned libraries. A good starting point to search for software is the Web location `http://www.netlib.org/`.

An algorithm is numerically stable if it does not introduce more sensitivity to perturbation than is already inherent in the problem. Stability ensures that the computed solution is near the solution of a slightly perturbed problem.

On the other hand, a problem is well conditioned if small changes in the data cause only small corresponding changes in the solution. If small changes in the data can potentially lead to large changes in the solution, the problem is ill-posed. This is crucial in applications, since first, we only have available measured data, which is prone to errors. And second, we can only represent numbers to within finite-precision arithmetic. If a problem is ill-conditioned, even if we use a numerically stable method to solve it, there is no guarantee that the result will be correct.

Consider for example the problem of determining whether the pair (A, B) is controllable. Controllability refers to the possibility of finding an input control over a prescribed lapse, transferring the system from any intial state to any final state. A well-known test for controllability says that (A, B) is controllable if and only if

$$\operatorname{rank}(\Gamma_c) \triangleq \operatorname{rank}(B, AB, \ldots, A^{(n-1)}B) = n$$

This is a numerically weak approach to study controllability, because the original problem (controllability) has been transformed to an intermediate problem which is far more sensible to perturbations. Consider the following classical example

$$A = \begin{bmatrix} 1 & 0 & & 0 \\ 0 & 2^{-1} & & 0 \\ & & \ddots & \\ 0 & 0 & & 2^{-9} \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

which is clearly controllable. If we use a digital computer with relative precision no smaller than 10^{-10}, then any good routine to determine rank (based on the singular value decomposition, to be explained later) would give as a result that $\operatorname{rank}(\Gamma_c)$ is smaller than 10. This is because the smallest singular values are 0.712×10^{-7}, 0.364×10^{-9} and 0.613×10^{-12}. The reader is warned against using the command `ctrb` of MATLAB for forming the controllability matrix of a system, for systems of order higher than 5.

Exercise

Can you think of alternative ways of computing the controllability properties of a system? Are they numerically more reliable? Have a look at what other possibilities MATLAB already offers.

A.2. Representation of linear model

Linear time-invariant models can be represented in a variety of forms. The state-space representation is a time domain representation, and is the most reliable type of model to use for computer analysis. It is the natural representation when problems arise from physical modelling with differential equations. We have developed the geometric theory almost exclusively from a time-domain perspective, which is an additional reason to use this type of representation. The advantages in the MIMO case are even greater. As a matter of fact, numerically stable routines to compute time responses (simulations), frequency responses, eigenvalues and other properties are readily available for this type of representation. Finite word-length in a computer should be the only concern for the practicioner. A well conditioned problem is usually a prerequisite for obtaining accurate results. Scaling is an important pre-process to consider.

The transfer function (matrix) representation is a frequency-domain representation of data. It is inherently ill-conditioned for a variety of reasons. Typically, large range of numbers and high order polynomials are one of the causes. The roots of polynomials are extremely sensitive to its coefficients. We should mention however, that most of the geometric results previously derived, have transfer function counterparts and that many control and estimation methods profitably use the so called *transfer function approach*.

A.2.1. Scaling. A well-conditioned problem is a prerequisite for obtaining accurate results. You should generally scale the (A, B, C, D) matrices of a system to improve their conditioning. The *condition number* of a matrix is the ratio of the largest to the smallest singular values (defined in the next section) of the matrix:

$$\kappa(A) \triangleq \sigma_1(A)/\sigma_n(A) \tag{A.1}$$

The condition number should ideally be close to unit. Its importance arises from the fact that each time we execute matrix multiplications (and these are ubiquitous in geometric control theory), the resulting quantities are more sensitive than the original ones (e.g. with respect to the solution of equations arising from the latter). Only orthogonal matrices have unity condition number. Matrices containing numbers widely spread in value are often poorly conditioned.

Scaling is also a prerequisite to infer certain *practical* properties of systems. After the introduction of the term "controllability" by Kalman, it has become synonymous of the mathematical notion of *state-controllability*, at least in the system theory community. Long before however, the term was employed to express the "ability of a process to achieve and maintain desired values" [Ziegler and Nichols]. This (input-output) controllability, is more in line with the engineers' feeling about the term: keep outputs within specified bounds or displacement from reference values, using available inputs and measurements, and despite the presence of disturbances and plant changes. It is clear that any meaningful statement about this other notion of controllability notion requires adequate scaling.

There are differing opinions as to how scaling should be done. In line with the previous practical concept, a common rule of thumb is to divide each variable by its maximum expected or allowed change. For the system

$$\begin{aligned}\dot{x} &= Ax + Bu\\ y &= Cx\end{aligned}$$

this is achieved by scaling each component input as

$$\hat{u}_i = u_i/u_i^{\max}$$

and similarly for outputs and (possibly) states. The overall effect of scaling is that of multiplying inputs, outputs and states by positive definite diagonal matrices D_1, D_2 and D_3 resulting in the system

$$\begin{aligned}\dot{\hat{x}} &= D_1 A D_1^{-1}\hat{x} + D_1 B D_2^{-1}\hat{u}\\ \hat{y} &= D_3 C D_1^{-1}\hat{x}\end{aligned}$$

where $\hat{x} = D_1 x$, $\hat{u} = D_2 u$, $\hat{y} = D_3 y$. With already stored data matrices A, B, C, no rounding errors are introduced if the D_i are integer powers of the floating point computer number base. Nondiagonal invertible matrices lead to general coordinate transformations. The engineer should choose the coordinate axes (e.g do coordinate transformations) and units (e.g. do scaling) so that the mathematical problem accurately reflects the sensitivity of the physical problem. A posteriori numerical conditioning can alter the sensitivity of the physically-conditioned problem, except for the case of orthogonal transformations.

A.3. The singular value decomposition

One of the most important tools of numerical analysis is the singular value decomposition (SVD) of a matrix. The SVD can be used to reliably compute many of the basic geometric objects of linear algebra, and has also found widespread use in many system-theoretic concepts such as balanced realizations, model truncation, induced norms for systems, etc. The presentation below is a purely theoretical one, and one should not attempt computing the SVD from the eigendecomposition indicated. There exist numerically stable routines to compute the SVD (e.g. the `svd` command of MATLAB), and this decomposition should be considered as an “elementary” operation upon which to build numerical algorithms, just as adding and multiplying numbers are elementary operations.

Theorem A.1. *Given a matrix A $m \times n$, there exist orthogonal matrices U $m \times m$ and V $n \times n$ such that*

$$A = U\Sigma V'$$

where

$$\Sigma = \begin{bmatrix} S & 0\\ 0 & 0\end{bmatrix}$$

and $S = diag(\sigma_1, \ldots, \sigma_r)$ with

$$\sigma_1 \geq \ldots \sigma_r > \sigma_{r+1} = \ldots \sigma_{\min\{m,n\}} = 0$$

These are the singular values.

For complex valued matrices, instead of transpose, conjugate transpose should be used, and *unitary* replaces *orthogonal* matrices.

PROOF

Since $A'A \quad n \times n$ is a symmetrical positive semidefinite matrix, it has non-negative eigenvalues σ_i^2, $i = 1, \ldots n$, that can be arranged so that $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r > 0 = \sigma_{r+1} = \ldots = \sigma_n$. Let $v_1, \ldots, v_n$ be a corresponding set of orthonormal eigenvectors (exists by the spectral decomposition theorem) and define

$$V \triangleq [\underset{r}{V_1} \quad \underset{n-r}{V_2}]$$

where

$$\begin{aligned} V_1 &= [v_1, \ldots, v_r] \\ V_2 &= [v_{r+1}, \ldots, v_n] \end{aligned}$$

With $S = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ and by definition of eigenvectors we have

$$A'AV_1 = V_1 S^2$$

from where

$$S^{-1}V_1'A'AV_1S^{-1} = I_r$$

Also $A'AV_2 = V_2 0_{n-r\times n}$ so that

$$V_2'A'AV_2 = 0$$

from where $AV_2 = 0$. Define $U_1 \triangleq AV_1S^{-1}$ thus obtaining $U_1'U_1 = I_r$. Finally, choose any U_2 such that

$$U \triangleq [\underset{r}{U_1} \quad \underset{m-r}{U_2}] \quad m \times m$$

is orthogonal (Gram-Schmidt procedure). Then we obtain

$$\begin{aligned} U'AV = \begin{bmatrix} U_1' \\ U_2' \end{bmatrix} A[V_1 V_2] &= \begin{bmatrix} U_1'AV_1 & U_1'AV_2 \\ U_2'AV_1 & U_2'AV_2 \end{bmatrix} \\ &= \begin{bmatrix} S & 0 \\ U_2'U_1S & 0 \end{bmatrix} \\ &= \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} \triangleq \Sigma \end{aligned}$$

∎

The numbers $\bar{\sigma}(A) \triangleq \sigma_1, \ldots, \underline{\sigma}(A) \triangleq \sigma_r, \ldots, \sigma_{\min\{m,n\}}$ are the singular values of A all of which are the positive square roots of the eigenvalues of $A'A$. The columns of U are called the *left singular vectors* of A while those of

V are called the *right singular vectors*. The matrix A' has the same singular values if this definition is adopted. The following easy relations hold:

$$\begin{array}{lcl} U'U & = & I_m \\ U_1'U_1 & = & I_r \\ U_2'U_2 & = & I_{m-r} \\ U_1'U_2 & = & 0_{r\times m-r} \\ UU' = U_1U_1' + U_2U_2' & = & I_m \end{array}$$

The *economy* size SVD is $A = U_1SV_1'$. We also have:

$$\begin{array}{lcl} Av_i & = & \sigma_i u_i \\ A'u_i & = & \sigma_i v_i \\ A & = & \sum_{i=1}^{r} \sigma_i u_i v_i' \end{array}$$

$$\begin{array}{lcl} \mathrm{rank}(A) & = & r \\ \ker(A) & = & \mathrm{span}\,\{v_{r+1},\dots,v_n\} \\ \mathrm{Im}\,(A) & = & \mathrm{span}\,\{u_1,\dots,u_r\} \end{array}$$

$$\|A\|_2 \triangleq \sup_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sigma_1 = \sigma_{\max}(A)$$

$$\|A\|_F \triangleq \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} \mid a_{ij} \mid^2} = \sqrt{\mathrm{trace}(A'A)} = \sqrt{\sum_{i=1}^{\min\{m,n\}} \sigma_i^2}$$

$$\begin{array}{rlcl} U_1U_1' = & \text{projection on to range}(A) & = & \Pi_c \\ U_2U_2' = & \text{projection on to range}(A)^\perp & = & \Pi_c^\perp = I_m - \Pi_c \\ V_1V_1' = & \text{projection on to kernel}(A)^\perp & = & \Pi_r \\ V_2V_2' = & \text{projection on to kernel}(A) & = & \Pi_r^\perp \end{array}$$

The SVD of A makes the various spaces associated with A explicit. So does any decomposition of A as $A = \bar{U}E\bar{V}'$ where $\bar{U}$ and $\bar{V}$ are any matrices whose columns span the column and row spaces of A. What makes SVD so useful is that the E obtained is diagonal, allowing for the expansion as a sum of rank-1 matrices u_iv_i' scaled by the σ_i (written above). Thus, the most important direction in the column space of A is u_1, with scale σ_1, and is reached by applying A to the unit-length vector v_1. The second most important is u_2 and so forth. This ranking of the directions leads naturally to optimal low-rank approximations of A. In fact, we have the following theorem:

THEOREM A.2. *Let $k < r = rank(A)$ and $A_k \triangleq \sum_{i=1}^{k} \sigma_i u_i v_i'$. Then*

$$\min_{rank(X)\leq k} \|A - X\|_2 = \|A - A_k\|_2 = \sigma_{k+1}$$

The proof is not difficult and is omitted.

The SVD also reveals the behaviour of the transformation A. Any vector x is first rotated in $\mathbb{R}^n$ by V' then scaled by the entries of Σ (where $n-r$ components are zeroed), and finally rotated in the $\mathbb{R}^m$ space by U to give Ax.

A.3.1. The numerical rank of a matrix. It is clear from the SVD decomposition, that the number of nonzero singular values of A determines the rank. The question is far more complicated in the context of finite precision arithmetics, but it is generally acknowledged that the SVD is the only reliable method of determining rank numerically. Extremely efficient and stable algorithms are available for computing the SVD. For these algorithms (e.g. `svd` in MATLAB), the computed singular values of A are in fact those of a *near* (within machine precision) matrix $A + E$.

When looking at the *smallest singular value* of a matrix, one should then consider the rank of all matrices in some δ neighbourhood around A, i.e. $\text{rank}(A + E) : \|E\| \leq \delta$. The choice of δ is done on the basis of the measurement errors to obtain A, or roundoff errors of previous computations. Theorem A.2 tells us then, that if we choose $\delta < \sigma_k$ we can consider A to have *numerical rank* k. The key quantity in rank determination is the (computed) σ_r. But since this number alone is scale-dependent, a better measure is $\sigma_r/\|A\| = \sigma_r/\sigma_1$, which is the reciprocal of the condition number of A with respect to pseudoinversion: $\kappa(A) = \|A\|/\|A^\dagger\|$. In case A is invertible this is the usual spectral condition number $\kappa(A) = \|A\|/\|A^{-1}\|$. Compare with (A.1).

A.3.2. Calculus of subspaces. Since the SVD is the only reliable way of computing rank, it follows that it is the only generally reliable way of computing the null and range subspaces of a matrix, and hence of the subspaces involved in geometric control theory. The reader should try writing general purpose routines for computing the following objects.

- Subspace inclusion. If R and T are two matrices with the same number of rows, checking whether $\text{Im } R \subset \text{Im } T$ is done upon verifying that $U_1 U_1' R = R$ where U_1 comes from the SVD of T. Prove this!
- The columns of U_1 from the SVD of the augmented matrix $[R\,T]$ are an orthonormal basis of $\text{Im } R + \text{Im } T$.
- To obtain a basis for the intersection of two subspaces, namely $\text{Im } R \cap \text{Im } T$ notice that if $x = [x_R'\, x_T']' \in \ker[R\,T]$ then $Rx_R = -Tx_T$.

A.4. A list of useful MATLAB commands

Following is a list of extremely useful MATLAB commands. The second segment requires the Control System Toolbox.

```
Matrix functions - numerical linear algebra.

Matrix analysis.
  cond        - Matrix condition number.
  norm        - Matrix or vector norm.
  rcond       - LINPACK reciprocal condition estimator.
  rank        - Number of linearly independent rows or columns.
```

```
  det         - Determinant.
  trace       - Sum of diagonal elements.
  null        - Null space.
  orth        - Orthogonalization.
  rref        - Reduced row echelon form.

Linear equations.
  \ and /     - Linear equation solution; use "help slash".
  chol        - Cholesky factorization.
  lu          - Factors from Gaussian elimination.
  inv         - Matrix inverse.
  qr          - Orthogonal-triangular decomposition.
  qrdelete    - Delete a column from the QR factorization.
  qrinsert    - Insert a column in the QR factorization.
  nnls        - Non-negative least-squares.
  pinv        - Pseudoinverse.
  lscov       - Least squares in the presence of known covariance.

Eigenvalues and singular values.
  eig         - Eigenvalues and eigenvectors.
  poly        - Characteristic polynomial.
  polyeig     - Polynomial eigenvalue problem.
  hess        - Hessenberg form.
  qz          - Generalized eigenvalues.
  rsf2csf     - Real block diagonal form to complex diagonal form.
  cdf2rdf     - Complex diagonal form to real block diagonal form.
  schur       - Schur decomposition.
  balance     - Diagonal scaling to improve eigenvalue accuracy.
  svd         - Singular value decomposition.

Matrix functions.
  expm        - Matrix exponential.
  expm1       - M-file implementation of expm.
  expm2       - Matrix exponential via Taylor series.
  expm3       - Matrix exponential via eigenvalues and eigenvectors.
  logm        - Matrix logarithm.
  sqrtm       - Matrix square root.
  funm        - Evaluate general matrix function.


Control System Toolbox.
Version 3.0b   3-Mar-93
Copyright (c) 1986-93 by The MathWorks, Inc.

Model building.
```

```
  append     - Append system dynamics.
  augstate   - Augment states as outputs.
  blkbuild   - Build state-space system from block diagram.
  cloop      - Close loops of system.
  connect    - Block diagram modeling.
  conv       - Convolution of two polynomials.
  destim     - Form discrete state estimator from gain matrix.
  dreg       - Form discrete controller/estimator from gain matrices.
  drmodel    - Generate random discrete model.
  estim      - Form continuous state estimator from gain matrix.
  feedback   - Feedback system connection.
  ord2       - Generate A,B,C,D for a second-order system.
  pade       - Pade approximation to time delay.
  parallel   - Parallel system connection.
  reg        - Form continuous controller/estimator from gain matrices.
  rmodel     - Generate random continuous model.
  series     - Series system connection.
  ssdelete   - Delete inputs, outputs, or states from model.
  ssselect   - Select subsystem from larger system.

Model conversions.
  c2d        - Continuous to discrete-time conversion.
  c2dm       - Continuous to discrete-time conversion with method.
  c2dt       - Continuous to discrete conversion with delay.
  d2c        - Discrete to continuous-time conversion.
  d2cm       - Discrete to continuous-time conversion with method.
  poly       - Roots to polynomial conversion.
  residue    - Partial fraction expansion.
  ss2tf      - State-space to transfer function conversion.
  ss2zp      - State-space to zero-pole conversion.

  tf2ss      - Transfer function to state-space conversion.
  tf2zp      - Transfer function to zero-pole conversion.
  zp2tf      - Zero-pole to transfer function conversion.
  zp2ss      - Zero-pole to state-space conversion.

Model reduction.
  balreal    - Balanced realization.
  dbalreal   - Discrete balanced realization.
  dmodred    - Discrete model order reduction.
  minreal    - Minimal realization and pole-zero cancellation.
  modred     - Model order reduction.

Model realizations.
  canon      - Canonical form.
```

```
  ctrbf      - Controllability staircase form.
  obsvf      - Observability staircase form.
  ss2ss      - Apply similarity transform.

Model properties.
  covar      - Continuous covariance response to white noise.
  ctrb       - Controllability matrix.
  damp       - Damping factors and natural frequencies.
  dcgain     - Continuous steady state (D.C.) gain.
  dcovar     - Discrete covariance response to white noise.
  ddamp      - Discrete damping factors and natural frequencies.
  ddcgain    - Discrete steady state (D.C.) gain.
  dgram      - Discrete controllability and observability gramians.
  dsort      - Sort discrete eigenvalues by magnitude.
  eig        - Eigenvalues and eigenvectors.
  esort      - Sort continuous eigenvalues by real part.
  gram       - Controllability and observability gramians.
  obsv       - Observability matrix.
  printsys   - Display system in formatted form.
  roots      - Polynomial roots.
  tzero      - Transmission zeros.
  tzero2     - Transmission zeros using random perturbation method.

Time response.
  dimpulse   - Discrete unit sample response.
  dinitial   - Discrete initial condition response.
  dlsim      - Discrete simulation to arbitrary inputs.
  dstep      - Discrete step response.
  filter     - SISO z-transform simulation.
  impulse    - Impulse response.
  initial    - Continuous initial condition response.
  lsim       - Continuous simulation to arbitrary inputs.
  ltitr      - Low level time response function.
  step       - Step response.
  stepfun    - Step function.

Frequency response.
  bode       - Bode plot (frequency response).
  dbode      - Discrete Bode plot (frequency response).
  dnichols   - Discrete Nichols plot.
  dnyquist   - Discrete Nyquist plot.
  dsigma     - Discrete singular value frequency plot.
  fbode      - Fast Bode plot for continuous systems.
  freqs      - Laplace-transform frequency response.
  freqz      - Z-transform frequency response.
```

```
  ltifr      - Low level frequency response function.
  margin     - Gain and phase margins.
  nichols    - Nichols plot.
  ngrid      - Draw grid lines for Nichols plot.
  nyquist    - Nyquist plot.
  sigma      - Singular value frequency plot.

Root locus.
  pzmap      - Pole-zero map.
  rlocfind   - Interactive root locus gain determination.
  rlocus     - Evans root-locus.
  sgrid      - Draw continuous root locus wn,z grid.
  zgrid      - Draw discrete root locus wn,z grid.

Gain selection.
  acker      - SISO pole placement.
  dlqe       - Discrete linear-quadratic estimator design.
  dlqew      - General discrete linear quadratic estimator design.
  dlqr       - Discrete linear-quadratic regulator design.
  dlqry      - Discrete regulator design with weighting on outputs.
  lqe        - Linear-quadratic estimator design.
  lqed       - Discrete estimator design from continuous cost function.
  lqe2       - Linear quadratic estimator design using Schur method.
  lqew       - General linear-quadratic estimator design.
  lqr        - Linear-quadratic regulator design.
  lqrd       - Discrete regulator design from continuous cost function.
  lqry       - Regulator design with weighting on outputs.
  lqr2       - Linear quadratic regulator design using Schur method.
  place      - Pole placement.

Equation solution.
  are        - Algebraic Riccati equation solution.
  dlyap      - Discrete Lyapunov equation solution.
  lyap       - Continuous Lyapunov equation solution.
  lyap2      - Lyapunov equation solution using diagonalization.

Demonstrations.
  ctrldemo   - Introduction to the Control Toolbox.
  boildemo   - LQG design of boiler system.
  jetdemo    - Classical design of jet transport yaw damper.
  diskdemo   - Digital control design of hard disk controller.
  kalmdemo   - Kalman filter design and simulation.
```

Bibliography

1. G. Basile and G. Marro, *Controlled and conditioned invariant subspaces in linear system theory*, J. Optimization Theory and Appl. **3** (1973), 306–316.
2. R.W. Brockett, *Asymptotic stability and feedback stabilization*, Differntial geometric control theory (1983), 181–191.
3. Jack Carr, *Applications of centre manifold theory*, Springer-Verlag, New York, 1981.
4. C.T. Chen, *Linear system theory and design*, HRW, New York, 1984.
5. J. Chen, *Sensitivity integral relations and design trade-offs in linear multivariable feedback systems.*, IEEE Transactions of Automatic Control **40** (1995), no. 10, 1700–1716.
6. B. Etkin, *Dynamics of atmospheric flight*, John Wiley & Sons, Inc., 1972.
7. B. Francis, *The linear multivariable regulator problem*, SIAM J. Control. Optim. **15** (1977), 486–505.
8. G. H. Golub and C. F Van Loan, *Matrix computations*, The Johns Hopkins University Press, 1993.
9. M. Hautus, *Linear matrix equations with applications to the regulator problem*, in Outils and Modeles, I.D. Landau, ed. (1983), 399–412.
10. X. Hu and C. Larsson, *On bounded peaking in the cheap control regulator*, Proceedings of the IFAC Youth Automation Conference (Beijing), 1996.
11. A. Isidori, *Nonlinear control systems*, Springer-Verlag, 1995.
12. M. Krasnoselskii and P. Zabreiko, *Geometric methods of nonlinear analysis*, springer-Verlag, Berlin, 1984.
13. A. S. Morse and W. M. Wonham, *Decoupling and pole assignment by dynamic compensation*, SIAM Journal on control and optimization **8** (1970), no. 3, 317–337.
14. L. M. Silverman and H. J. Payne, *Input-output structure of linear systems with applications to the decoupling problem*, SIAM Journal on Control and Optimization **9** (1971), no. 2, 199–233.
15. H. Trentelman, A. Stoorvogel, and M. Hautus, *Control theory for linear systems*, Springer-Verlag, 2001.
16. J.C. Willems and C. Commault, *Disturbance decoupling by measurement feedback with stability or pole placementy ity o*, SIAM Journal of Control and Optimization **19** (1981), 490–504.
17. W. M. Wonham, *Linear multivariable control: a geometric approach*, Springer-Verlag, 1979.

Index