Numerical Algorithms in Control

Z. Drmač

Abstract

Control theory poses interesting and challenging problems to numerical mathematics, in particular to matrix theory and numerical linear algebra. Modern theoretical developments and exciting engineering applications demand efficient and numerically sound algorithms implemented as robust and accurate numerical software. Better understanding of the sensitivity of numerical problems, and new paradigms in the algorithmic development open new possibilities and allow high accuracy solutions to problems that are usually deemed numerically intractable. In this series of lectures, we show how some recent developments in accurate linear algebra (accurate algorithms for eigenvalues and singular values, and corresponding theory) improve numerical computations in control theory, and contribute to software improvements.

We will use few separate topics as case studies. They are divided in two groups: *(i)* construction and analysis of numerical algorithms; *(ii)* numerical software development.

(i) We show that carefully designed algorithms (based on detailed error analysis and perturbation theory) can e.g. simultaneously diagonalize a pair of positive definite matrices (e.g. for balancing the grammians of a LTI system) and compute the Hankel singular values with small backward relative perturbations of matrix entries. This is much stronger stability than computing with backward error that is small in the matrix norm sense. Then, a stronger perturbation theory will identify better condition numbers and guarantee higher accuracy. As our second example, we consider computational tasks (e.g. least squares rational approximations) where the underlying matrices have a generalized Cauchy or Vandermonde structure. Such matrices are notoriously ill-conditioned, but with carefully designed algorithms all computation with them can be done very accurately – we show the details. We use other examples (computing certain canonical forms in control, model order reduction algorithms) to show how some instabilities undetected spoil the accuracy, and that they are removable by modifications inspired by error analysis and perturbation theory.

(ii) Advanced applications are based on high level packages, such as Matlab, and computing engines such as numerical software libraries LAPACK, SLICOT. We stress the importance of reliable numerical software and call for more mathematical rigor in the implementation phase (coding) and testing. To illustrate

a problem, we show how adding just one " `WRITE(*,*) variable`" statement to a mission critical code based on the above mentioned libraries, or changing compiler options, completely changes the computed key parameters of a given linear time invariant (LTI) system. Such situations may occur only at certain distance to singularity, and some computational tasks (such as e.g. revealing a numerical rank) are usually performed and are crucial (and interesting) on data close to singularity. And, since many phenomena are possible when close to singularity, any ill–behavior of the software is usually attributed to ill–conditioning, bailed out by backward stability, and the true problem may remain inconspicuous. (We give an example of rank revealing QR factorization software (LINPACK, LAPACK, SLICOT,...) instability that had been circulating undetected in all relevant matrix computation libraries for more than thirty years.) This is certainly undesired behavior, even if such computation remains backward stable, and even if the computation is doomed to fail, due to ill–conditioning. We show the advantages of using state of the art matrix perturbation theory in rigorous numerical linear algebra software development.

All algorithms described in the theoretical lectures will be implemented (e.g. in Matlab) in the lab exercises.