Computer-Aided Control Systems Design: Introduction and Historical Overview

Andreas Varga
German Aerospace Center, DLR Oberpfaffenhofen
Institute of System Dynamics and Control
Münchnerstr. 20, 82234 Wessling, Germany
Phone: +49 (8153) 282407
Fax: +49 (8153) 281850
Andreas.Varga@dlr.de

Abstract

Computer-aided control system design (CACSD) encompasses a broad range of methods, tools and technologies for system modelling, control system synthesis and tuning, dynamic system analysis and simulation, as well as validation and verification. The domain of CACSD enlarged progressively over decades from simple collections of algorithms and programs for control system analysis and synthesis, to comprehensive tool sets and user-friendly environments supporting all aspects of developing and deploying advanced control systems in various application fields. This article gives a brief introduction to CACSD and reviews the evolution of key concepts and technologies underlying the CACSD domain. Several cornerstone achievements in developing reliable numerical algorithms, implementing robust numerical software and developing sophisticated integrated modelling, simulation and design environments are highlighted.

Keywords and Phrases

CACSD, simulation, modeling, numerical analysis, software tools

Introduction

To design a control system for a plant, a typical *computer-aided control system design* (CACSD) work flow comprises several interlaced activities.

Model building is often a necessary first step consisting in developing suitable mathematical models to accurately describe the plant dynamical behaviour. High-fidelity physical plant models, obtained for example by using first principles modelling, primarily serve for analysis and validation purposes using appropriate simulation techniques. These dynamic models are usually defined by a set of *ordinary differential equations* (ODEs), *differential-algebraic equation* (DAEs) or *partial differential equations* (PDEs). However, for control system synthesis purposes simpler models are required, which are derived by simplifying high-fidelity models (e.g., by linearization, discretization, or model reduction) or directly determined in a specific form from input-output measurement data using system identification techniques. Frequently used synthesis models are continuous or discrete-time *linear time-invariant* (LTI) models describing the nominal behaviour of the plant in a specific operating point. The more accurate *linear parameter varying* (LPV) models may serve to account for uncertainties due to various performed approximations, nonlinearities or varying model parameters.

Simulation of dynamical systems is a closely related activity to modelling and is concerned with performing virtual experiments on a given plant model to analyse and predict the dynamic behaviour of a physical plant. Often modelling and simulation are closely connected parts of dedicated environments, where specific classes of models can be built and appropriate simulation methods can be employed. Simulation is also a powerful tool for the validation of mathematical models and their approximations. In the context of CACSD, simulation is frequently used as a control system tuning-aid, as, for example, in an optimization-based tuning approach using time-domain performance criteria.

System analysis and synthesis are concerned with with the investigation of properties of the underlying synthesis model and in the determination of a control system which fulfills basic requirements for the closed-loop controlled plant, such as stability or various time or frequency response requirements. The analysis also serves to check existence conditions for the solvability of synthesis problems, according to established design methodologies. An important synthesis goal is the guarantee of the performance robustness. To achieve this goal robust control synthesis methodologies often employ optimization-based parameter tuning in conjunction with worst-case analysis techniques. A rich collection of reliable numerical algorithms are available to perform such analysis and synthesis tasks. These algorithms form the core of CACSD and their development represented one of the main motivations for CACSD-related research.

Performance robustness assessment of the resulting closed-loop control system is a key aspect of the verification and validation activity. For a reliable assessment, simulation-based worst-case analysis represents often the only way to prove the performance robustness of the synthesized control system in the presence of parametric uncertainties and variabilities.

Development of CACSD tools

The development of CACSD tools for system analysis and synthesis started around 1960, immediately after general purpose digital computers and new programming languages became available for research and development purposes. In what follows we give a historical survey of these developments in the main CACSD areas.

Modelling and simulation tools

Among the first developments were modelling and simulation tools for continuous-time systems described by differential equations based on dedicated simulation languages. Over forty continuous-system simulation languages had been developed as of 1974 [Nilsen and Karplus, 1974], which evolved out of attempts at digitally emulating the behaviour of widely used analog computers before 1960. A notable development in this period was the CSSL standard [Augustin et al, 1967], which defined a system as an interconnection of blocks corresponding to operators which emulated the main analog simulation blocks and implied the integration of the underlying ODEs using suitable numerical methods. For many years, the ACSL preprocessor to Fortran [Mitchel and Gauthier, 1976] was one of the most successful implementations of the CSSL-standard.

A turning point was the development of graphical user interfaces allowing graphical block-diagram based modelling. The most important developments were SystemBuild [Shah et al, 1985] and SIMULAB (later marketed as Simulink) [Grace, 1991]. Both products used a customizable set of block libraries and were seamlessly integrated in, respectively, MATRIXx and MATLAB, two powerful interactive matrix computation environments (see below). SystemBuild provided several advanced features such as event management, code generation, and DAE-based modelling and simulation. Simulink excelled from the beginning with its intuitive, easy-to-use user interface. Recent extensions of Simulink allow the modelling and simulation of hybrid systems, code generation for real-time applications, and various enhancements of the model building process (e.g., object-oriented modelling).

The object-oriented paradigm for system modelling was introduced with Dymola [Elmqvist, 1978] to support physical system modelling based on interconnections of subsystems. The underlying modelling language served as the basis of the first version of Modelica [Elmquist et al., 1997], a modern equation-based modeling language which was the result of a coordinated effort for the unification and standardization of expertise gained over many years with object-oriented physical modelling. The latest developments in this area are comprehensive model libraries for different application domains such as mechanical, electrical, electronic, hydraulic, thermal, control, and electric power systems. Notable commercial front-ends for Modelica are Dymola, MapleSim and SystemModeler, where the last two are tightly integrated in the symbolic computation environments Maple and Mathematica, respectively.

Numerical software tools

The computational tools for CACSD rely on many numerical algorithms whose development and implementation in computer codes was the primary motivation of this research area since its beginnings. The Automatic Synthesis Program (ASP) developed in 1966 [Kalman and Englar, 1966], was implemented in FAP (Fortran Assembly Program) and ran only on an IBM 7090-7094 machine. The Fortran II version of ASP (known as FASP) can be considered to be the first collection of computational CACSD tools ported to several mainframe computers. Interestingly, the linear algebra computations were covered by only three routines (diagonal decomposition, inversion and pseudo-inverse) and no routines were used for eigenvalue or polynomial roots computation. The main analysis and synthesis functions covered the sampled-data discretization (via matrix exponential), minimal realization, time-varying Riccati equation solution for quadratic control, filtering, and stability analysis. The FASP program itself performed the required computational sequences by interpreting simple commands with parameters. The extensive documentation containing a detailed description of algorithmic approaches and many examples marked the starting point of an intensive research on algorithms and numerical software, which culminated in the development of the high-performance control and systems library SLICOT [Benner et al, 1999; Huffel et al, 2004]. In what follows, we highlight the main achievements along this development process.

The direct successor of FASP is the Variable Dimension Automatic Synthesis Program (VASP) (implemented in Fortran IV on IBM 360) [White and Lee, 1971], while a further development was ORACLS [Armstrong, 1978], which included several routines from the newly developed eigenvalue package EISPACK [Smith et al, 1976; Garbow et al, 1977] as well as solvers for linear (Lyapunov, Sylvester) and quadratic (Riccati) matrix equations. From this point, the mainstream development of numerical algorithms for linear system analysis and synthesis closely followed the development of algorithms and software for numerical linear algebra. A common feature of all subsequent developments was the extensive use of robust linear algebra software with the Basic Linear Algebra Subprograms (BLAS) [Lawson et al, 1979] and the Linear Algebra Package (LINPACK) for solving linear systems [Dongarra et al, 1979]. Several control libraries have been developed almost simultaneously, relying on the robust numerical linear algebra core software formed of BLAS, LINPACK and EISPACK. Notable examples are: RASP (based partly on VASP and ORACLS) [Grübel, 1983] – developed originally by the University of Bochum and later by the German Aerospace Center (DLR); BIMAS [Varga and Sima, 1985] and BIMASC [Varga and Davidoviciu, 1986] – two Romanian initiatives; and SLICOT [Boom et al, 1991] – a Benelux initiative of several universities jointly with the Numerical Algorithm Group (NAG).

The last development phase was marked by the availability of the Linear Algebra Package (LAPACK) [Anderson et al, 1992], whose declared goal was to make the widely used EISPACK and LINPACK libraries run efficiently on shared-memory vector and parallel processors. To minimize the development efforts, several active research teams from Europe started, in the framework of the NICONET Project, a concentrated effort to develop a high-performance numerical software library for CACSD as a new significantly extended version of the original SLICOT. The goals of the new library were to cover the main computational needs of CACSD, by relying exclusively on LAPACK and BLAS, and to guarantee similar numerical performance as that of the LAPACK routines. The software development used rigorous standards for implementation in Fortran 77, modularization, testing, and documentation (similar to that used in LAPACK). The development of the latest versions of RASP and SLICOT eliminated practically any duplication of efforts and led to a library which contained the best software from RASP, SLICOT, BIMAS and BIMASC. The current version of SLICOT is fully maintained by the NICONET Association[1] and serves as basis for implementing advanced computational functions for CACSD in interactive environments as MATLAB[2], Maple[3], Scilab[4] and Octave[5].

Interactive tools

Early experiments during 1970–1985 included the development of several interactive CACSD tools employing menu-driven interaction, question–answer dialogues or command languages. The April 1982 special issue of IEEE Control Systems Magazine was dedicated to CACSD environments and presented software summaries of 20 interactive CACSD packages. This development period was marked by the establishment of new standards for programming languages (Fortran 77, C), availability of high-quality numerical software libraries (BLAS, EISPACK, LINPACK, ODEPACK), transition from mainframe computers to minicomputers and finally to the nowadays-ubiquitous personal computers as computing platforms, spectacular developments in graphical display technologies, and application of sound programming paradigms (e.g., strong data typing).

A remarkable event in this period was the development of MATLAB, a command language based interactive matrix laboratory [Moler, 1980]. The original version of MATLAB was written in Fortran 77. It was primarily intended as a student teaching tool and provided interactive access to selected subroutines from LINPACK and EISPACK. The tool circulated for a while in the public domain and its high flexibility was soon recognized. Several CACSD-oriented commercial clones have been implemented in the C language, the most important among them being MATRIXx [Walker et al, 1982] and PC-MATLAB [Moler et al, 1985].

The period after 1985 until around 2000 can be seen as a consolidation and expansion period for many commercial and non-commercial tools. In an inventory of CACSD-related software issued by the Benelux Working Group on Software (WGS) under

[1]http://www.niconet-ev.info/en/
[2]http:// www.mathworks.com
[3]http://www.maplesoft.com/products/maple/
[4]http://www.scilab.org/
[5]http://www.gnu.org/software/octave/

the auspices of the IEEE Control Systems Society, there were in 1992 in active development 70 standalone CACSD packages, 21 tools based on or similar to MATLAB, and 27 modelling/simulation environments. It is interesting to look more closely at the evolutions of the two main players MATRIXx and MATLAB, which took place under harshly competitive conditions.

MATRIXx with its main components Xmath, SystemBuild and AutoCode, had over many years a leading role (especially among industrial customers), excelling with a rich functionality in domains such as system identification, control system synthesis, model reduction, modelling, simulation, code generation. After 2003, MATRIXx[6] became a product of National Instruments Corporation and complements its main product family LabView, a visual programming language based system-design platform and development environment[7].

MATLAB gained broad academic acceptance by integrating many new methodological developments in the control field into several control-related toolboxes. MATLAB also evolved as a powerful programming language, which allows easy object-oriented manipulation of different system descriptions via operator overloading. At present, the CACSD tools of MATLAB and Simulink represent the industrial and academic standard for CACSD. The existing CACSD tools are constantly extended and enriched with new model classes, new computational algorithms (e.g., structure-exploiting eigenvalue computations based on SLICOT), dedicated graphical user interfaces (e.g., tuning of PID controllers or control-related visualizations), advanced robust control system synthesis, etc. Also many third-party toolboxes contribute to the wide usage of this tool.

Basic CACSD functionality incorporating symbolic processing techniques and higher precision computations is available in the Maple product MapleSim Control Design Toolbox as well as in the Mathematica Control Systems product. Free alternatives to MATLAB are the MATLAB-like environments Scilab, a French initiative pioneered by INRIA, and Octave, which has recently added some CACSD functionality.

Summary and Future Directions

The development and maintenance of integrated CACSD environments, which provide support for all aspects of the CACSD cycle such as modelling, design, and simulation, involve sustained, strongly interdisciplinary efforts. Therefore, the CACSD tool development activities must rely on the expertise of many professionals covering such diverse fields as control system engineering, programming languages and techniques, man-machine interaction, numerical methods in linear algebra and control, optimization, computer visualization, and model building techniques. This may explain why currently, only a few of the commercial developments of prior years are still in use and actively maintained/developed. Unfortunately, the number of actively developed non-commercial alternative products is even lower. The dominance of MATLAB, as a *de facto* standard for both industrial and academic usage of integrated tools covering all aspects of the broader area of *computer aided control engineering* (CACE), can not be overseen.

The new trends in CACSD are partly related to handling more complex applications, involving time-varying (e.g., periodic, multi-rate sampled-data, and differential-algebraic) linear dynamic systems, nonlinear systems with many parametric uncertainties, and large-scale models (e.g., originating from the discretization of PDEs). To address many computational aspects of model building (e.g., model reduction of large order systems), optimization-based robust controller tuning using multiple-model approaches, or optimization-based robustness assessment using global-optimization techniques, parallel computation techniques allow substantial savings in computational times and facilitate addressing computational problems for large scale systems. A topic which needs further research is the exploitation of the benefits of combining numerical and symbolic computations (e.g., in model building and manipulation).

Cross-references

CACSD software tools, modelling and simulation, system identification, model reduction, optimization-based design, robust synthesis, robustness assessment, verification and validation

Recommended reading

The historical development of CACSD concepts and techniques was the subject of several articles in reference works [Rimvall and Jobling, 1995; Schmid, 2002]. A selection of papers on numerical algorithms underlying the development of CACSD appeared in [Patel et al, 1994]. The special issue No. 2/2004 of the IEEE Control Systems Magazine on *Numerical Awareness in Control* presents several surveys on different aspects of developing numerical algorithms and

[6]http://www.ni.com/matrixx/
[7]http://www.ni.com/labview

software for CACSD.

The main trends over the last three decades in CACSD related research can be followed in the programs/proceedings of the biannual IEEE Symposia on CACSD from 1981 to 2013 (partly available at http://ieeexplore.ieee.org) as well as of the triennial IFAC Symposia on CACSD from 1979 to 2000. Additional information can be found in several CACSD-focused survey articles and special issues (e.g., No. 4/1982; No. 2/2000) of the IEEE Control Systems Magazine.

Anderson E, Bai Z, Bishop J, Demmel J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Ostrouchov S, Sorensen D (1992) LAPACK User's Guide. SIAM, Philadelphia

Armstrong ES (1978) ORACLS – A System for Linear-Quadratic Gaussian Control Law Design. Technical Paper 1106 96-1, NASA

Augustin DC, Strauss JC, Fineberg MS, Johnson BB, Linebarger RN, Sansom FJ (1967) The SCi continuous system simulation language (CSSL). Simulation 9:281–303

Benner P, Mehrmann V, Sima V, Van Huffel S, Varga A (1999) SLICOT – a subroutine library in systems and control theory. In: Datta BN (ed) Applied and Computational Control, Signals and Circuits, vol 1, Birkhäuser, pp 499–539

Boom A, Brown A, Geurts A, Hammarling S, Kool R, Vanbegin M, Van Dooren P, Huffel SV (1991) SLICOT, a subroutine library in control and systems theory. In: Prepr. 5th IFAC/IMACS Symp. CADCS'91, Swansea, UK, Pergamon Press, Oxford, pp 89–94

Dongarra JJ, Moler CB, Bunch JR, Stewart GW (1979) LINPACK User's Guide. SIAM Publications, Philadelphia

Elmquist H et al (1997) Modelica – A Unified Object-Oriented Language for Physical Systems Modeling (Version 1). URL http://www.modelica.org/documents/Modelica1.pdf

Elmqvist H (1978) A structured model language for large continuous systems. PhD thesis, Department of Automatic Control, Lund University, Sweden

Garbow BS, Boyle JM, Dongarra JJ, Moler CB (1977) Matrix Eigensystem Routines – EISPACK Guide Extension. Springer-Verlag, Heidelberg

Grace ACW (1991) SIMULAB, an integrated environment for simulation and control. In: Proc. of American Control Conference, Boston, MA, USA, pp 1015–1020

Grübel G (1983) Die regelungstechnische Programmbibliothek RASP. Regelungstechnik 31:75–81

Huffel SV, Sima V, Varga A, Hammarling S, Delebecque F (2004) High-performance numerical software for control. Control Systems Magazine 24:60–76

Kalman RE, Englar TS (1966) A User's Manual for the Automatic Synthesis Program (Program C). Technical Report CR-475, NASA

Lawson CL, Hanson RJ, Kincaid DR, Krogh FT (1979) Basic linear algebra subprograms for Fortran usage. ACM Transactions on Mathematical Software 5:308–323

Mitchel EEL, Gauthier JS (1976) Advanced continuous simulation language (ACSL). Simulation 26:72–78

Moler CB (1980) MATLAB Users' Guide. Tech. rep., Department of Computer Science, University of New Mexico, Albuquerque, USA

Moler CB, Little J, Bangert S, Kleinman S (1985) PC-Matlab, Users'Guide, Version 2.0. Tech. rep., The MathWorks, Inc, 158 Woodland St, Sherborn, MA USA

Nilsen RN, Karplus WJ (1974) Continuous-system simulation languages: A state-of-the-art survey. Mathematics and Computers in Simulation 16:17–25, DOI http://dx.doi.org/10.1016/S0378-4754(74)80003-0

Patel RV, Laub AJ, Van Dooren (Eds) P (1994) Numerical Linear Algebra Techniques for Systems and Control. IEEE Press, Piscataway, NJ, USA

Rimvall C, Jobling CP (1995) Computer-Aided Control Systems Design. In: Levine WS (ed) The CONTROL HANDBOOK, CRC Press, pp 429–442

Schmid C (2002) Computer-Aided Control System Engineering Tools. In: Unbehauen H (ed) Control Systems, Robotics and Automation, Encyclopedia of Life Support Systems (EOLSS), URL http://www.eolss.net

Shah CS, Floyd MA, Lehman LL (1985) MATRIXx: Control design and model building CAE capabilities. In: Jamshidi M, Herget CJ (eds) Advances in Computer Aided Control Systems Engineering, North-Holland, Elsevier Science Publishers, Amsterdam, pp 181–207

Smith BT, Boyle JM, Dongarra JJ, Garbow BS, Ikebe Y, Klema VC, Moler CB (1976) Matrix Eigensystem Routines - EISPACK Guide, Second Edition, Lecture Notes in Computer Science, vol 6. Springer-Verlag

Varga A, Davidoviciu A (1986) BIMASC - A package of Fortran subprograms for analysis, modelling, design and simulation of control systems. In: Hansen NE, Larsen PM (eds) Prepr. of 3rd IFAC/IFIP Int. Symposium on Computer Aided Design in Control and Engineering (CADCE'85), Copenhagen, Denmark, Pergamon Press, Oxford, pp 151–156

Varga A, Sima V (1985) BIMAS - A basic mathematical package for computer aided systems analysis and design. In: Gerter J, Keviczky L (eds) Prepr. of 9th IFAC World Congress, Budapest, Hungary, vol 8, pp 202–207

Walker R, Gregory C, Shah S (1982) MATRIXx: A data analysis, system identification, control design and simulation package. Control Systems Magazine 2:30–37

White JS, Lee HQ (1971) Users Manual for the Variable Automatic Synthesis Program (VASP). Technical Memorandum TM X-2417, NASA