NICONET
Ein europäisches Netzwerk zur Numerik in der Kontrolltheorie

Peter Benner*

Zentrum für Technomathematik
Fachbereich 3 – Mathematik und Informatik
Universität Bremen
28334 Bremen

E-mail: benner@math.uni-bremen.de
URL: http://www.numerik.uni-bremen.de/~benner

17. März 1998

Zusammenfassung

Es wird ein Überblick gegeben über eine europäische Initiative zur Erstellung bzw. Weiterführung einer Softwarebibliothek (SLICOT) für numerische Berechnungen in der Kontroll- (Regelungs-, Steuerungs-) und Systemtheorie. Anhand von Beispielen wird demonstriert, daß die numerischen Algorithmen oder aber deren Implementierung in CACSD Umgebungen oft unbrauchbar sind. Mit Hilfe von SLICOT sollen numerisch robuste Verfahren auch in CACSD Entwicklungsumgebungen zur Verfügung gestellt werden. Dazu haben die Entwickler von SLICOT — die Working Group on Software (WGS) — eine Initiative für ein europäisches Netzwerk gestartet, welches unter dem Namen NICONET die Entwicklung der Bibliothek und deren Integration in CACSD Entwicklungsumgebungen vorantreiben soll.

*Als Vertreter der Working Group on Software – WGS.

Überblick

- Einführung
- Warum muß man über Numerik in der Kontrolltheorie reden?
- Entwicklung einer Softwarebibliothek für System- und Kontrolltheorie — WGS
- SLICOT
- NICONET
- Ausblick

Probleme bei der praktischen Anwendung vieler theoretischer Methoden:

Die Entwicklung von in der Praxis anwendbarer Software sollte den modernen Entwicklungen in

- der mathematischen Theorie,
- den Anwendungen,
- den algorithmischen Verfahren

für kontroll- und systemtheoretische Probleme Rechnung tragen.
Zur Simulation und zur (hard- und/oder softwaretechnischen) Implementierung bzw. Realisierung sind in der Regel numerische Verfahren erforderlich (selbst wenn mit Computeralgebra–Systemen gearbeitet wird!), insbes. in den meisten CACSD (COMPUTER-AIDED CONTROL SYSTEM DESIGN) Entwicklungsumgebungen.
Reale Probleme sind oft

- *hochdimensional*,
- *schlecht gestellt*,
- *schlecht skaliert* (was nicht mit schlecht gestellt oder schlecht konditioniert gleichzusetzen ist) — insbesondere durch die unreflektierte Benutzung von SI–Einheiten (siehe auch Beispiel 2).

Software sollte deshalb *zuverlässig*, *numerisch robust*, und *effizient* sein.
Folgende Punkte sollten daher beachtet werden:

- Hohe Dimensionen können durch Ausnutzung vorhandener Strukturen behandelt werden (z.B. Ausnutzung von dünner Besetzung von Matrizen, etc.).
- Numerische Robustheit fordert
  - automatische Skalierung,
  - Ausgabe einer Warnung bei schlecht gestellten/konditionierten Problemen,
  - möglichst genaue Resultate,
  - Ausgabe sinnvoller Resultate, so daß sie dem Problem zugeordnet werden können: Ist z.B. die Lösung eine symmetrische Matrix, so sollte auch eine symmetrische Matrix als numerische Lösung berechnet werden, da sonst ggfs. die theoretischen Grundlagen nachfolgender Rechnungen verletzt sein könnten.
- Evaluierung sollte anhand standardisierter Benchmark-Beispiele erfolgen.

Warum reicht MATLAB [19] nicht aus?

(oder eine andere integrierte Entwicklungsumgebung wie Scilab [8], ANDECS [13], EASY5 [7], XMath [15],...?)
Matlab

- ist benutzerfreundlich,
- besitzt einen Kern von robusten mathematischen Verfahren,
- ist ein ausgezeichnetes Hilfsmittel zur Entwicklung neuer algorithmischer Verfahren,
- wird intensiv eingesetzt in Forschung, Lehre und auch Industrie (z.B. bei ABB, Philips, Ford, AspenTech,...)

Aber:

- Der Kern ist z.T. veraltet ($\longrightarrow$ *langsam*)
- Der Grunddatentyp ist die dichtbesetzte, komplexe Matrix $\longrightarrow$
  - nur "kleine" Systeme können behandelt werden,
  - langsam, da komplexe Arithmetik 4–6 mal teurer als reelle Berechnungen.
- Zur Ausnutzung von Struktur müßen eigene Matlab-Funktionen geschrieben werden. Diese werden interpretiert, was wiederum eine langsame Ausführungsgeschwindigkeit bedeutet.
- Z.T. sind in den Matlab Toolboxen schlechte Algorithmen implementiert, Implementierungen selbst sind oft schlecht, insbesondere in der

  Control Toolbox [17], Robust Control Toolbox [18], u.v.a.

  Siehe die folgenden Beispiele.

Beispiel 1:

Betrachte den Löser `care` für *algebraische Riccatigleichungen* (ARE) der Form

$$0 \;=\; C^TQC + A^TX + XA - XBR^{-1}B^TX$$

aus der MATLAB Control Toolbox [17] als auch der LMI Toolbox [11]. Dieser Löser basiert auf der Lösung des zugehörigen verallgemeinerten Eigenwertproblems mit dem QZ-Algorithmus; siehe [2, 26]. In `care` wird dazu die interne MATLAB-Funktion `qz` (komplexer QZ-Algorithmus aus EISPACK [20], 1976) benutzt. Für Testzwecke wurde die Funktion `qz` ersetzt durch die moderne LAPACK-Implementierung [1] (durch Einbinden des Fortran 77 Codes mit Hilfe eines MEX–Files).
Tabelle 1 gibt einen Vergleich der CPU–Zeiten von `care` und der modifizierten Version `care_new`. Beachte: Die gesuchte Lösung X der ARE ist für ein n–dimensionales System eine symmetrische $n \times n$–Matrix, d.h., es handelt sich um ein nichtlineares Gleichungssystem mit $\approx n^2/2$ Unbekannten! (Z.B. für $n = 100$ also ca. 5000 Unbekannte.)

Tabelle 1: Vergleich der CPU-Zeiten (Pentium–PC, 133 MHz)

n	`care`	`care_new`	Speed-Up
20	1.71 sec.	0.38 sec.	4.5
60	42.95 sec.	9.01 sec.	4.8
120	303.00 sec.	63.40 sec.	4.8

Nur durch Ersetzen veralteter Implementierungen (und damit Benutzung von reeller Arithmetik und modernen block–orientierten Algorithmen) reduziert sich hier also die Ausführungszeit auf ca. ein Fünftel!

Beispiel 2: (Eurocopter/DLR[1])[2]

Das physikalische Modell eines Aktors für die aktive Vibrationsdämpfung eines Hubschraubers führt auf ein lineares, zeitinvariantes System der Form

$$
\begin{aligned}
\dot{x} &= Ax + Bu, \qquad x \in \mathbb{R}^5, \quad u \in \mathbb{R}^2, \\
y &= Cx + Du, \qquad y \in \mathbb{R}^5.
\end{aligned}
$$

Das System ist vollständig steuerbar und beobachtbar!
Die Koeffizientenmatrizen sind gegeben durch $D = 0 \in \mathbb{R}^{5\times 2}$ und

```
A =

            0   1.0000e+00            0            0            0
  -1.5800e+06  -1.2570e+03            0            0            0
   3.5410e+14            0  -1.4340e+03            0  -5.3300e+11
            0            0            0            0   1.0000e+00
            0            0            0  -1.8630e+04  -1.4820e+00


B =

            0            0
   1.1030e+02            0
            0            0
            0            0
            0   8.3330e-03


C =

   1.0000e+00            0            0            0            0
            0            0   1.0000e+00            0            0
            0            0            0   1.0000e+00            0
   6.6640e-01            0  -6.2000e-13            0            0
            0            0  -1.0000e-03   1.8960e+06   1.5080e+02
```

Zunächst überprüfen wir die Steuerbarkeit: Die Funktion `ctrbf` aus der MATLAB Control Toolbox [17] berechnet einen Vektor `k`, dessen Komponenten ganze Zahlen sind und deren Summe die Dimension des steuerbaren Unterraums angibt.

[1] Deutsches Zentrum für Luft- und Raumfahrt e.V.
[2] Siehe auch `http://www.op.dlr.de/FF-DR-ER/research/num/scaling.html`

```
>> [aa,bb,cc,t,k]=ctrbf(A,B,C); sum(k)

ans =

     3
```

D.h., MATLAB hält das System für nicht steuerbar!
Ein noch merkwürdigereres Ergebnis erhält man, wenn man versucht, eine minimale Realisierung mit Hilfe der MATLAB Control Toolbox Funktion `minreal` [17] auszurechnen.

```
>> [aa,bb,cc,dd]=minreal(A,B,C,D)
5 state(s) removed

aa =

     []


bb =

   Empty matrix: 0-by-2


cc =

   Empty matrix: 5-by-0


dd =

     0     0
     0     0
     0     0
     0     0
     0     0
```

Alle Zustandsraumvariablen wurden entfernt — obwohl das System vollständig beobachtbar und vollständig steuerbar ist.
Das Problem hierbei ist die schlechte Skalierung der Systemmatrizen durch Benutzung verschiedener SI–Einheiten. Mit einer einfachen Umskalierung kann man dieses Problem umgehen. Solche Skalierungen sollten also für robuste numerische Software Standard sein!

Ziel

Softwarebibliothek numerisch robuster, zuverlässiger, effizienter Algorithmen für grundlegende Berechnungen in der System- und Kontrolltheorie, und zwar

- unabhängig von CACSD Entwicklungsumgebungen,
- portabel ($\Longrightarrow$ Fortran oder C)
- integrierbar in möglichst viele Entwicklungsumgebungen,
  dabei sollte der Nutzer nicht merken, daß es sich nicht um eingebaute Funktion handelt, so daß für den Nutzer keine Fortran– bzw. C–Kenntnisse nötig sind;
- basierend auf anderen robusten und effizienten mathematischen Softwarebibliotheken wie

  BLAS [16, 10, 9] und LAPACK [1]

Z.Zt. werden zwei solche Bibliotheken weiterentwickelt:

- Die **S**ubroutine **L**ibrary **i**n **S**ystems and **Co**ntrol **T**heory, kurz SLICOT, von der **W**orking **G**roup on **S**oftware, kurz WGS, siehe [5, 21, 22, 24, 25, 28], und
- die regelungstechnische Programmbibliothek RASP vom DLR [12].

The Working Group on Software (WGS)

- Start: Anfang der 80er Jahre in den **BeNe**Lux-Staaten.
- Ziel: Zusammenführung existierender numerischer Software und
  Einbringung derselben in eine Softwarebibliothek
- Zusammenarbeit mit NAG[3] in den späten 80er Jahren.
- 1. Version von SLICOT 1990 [21].
- Kooperation mit der DLR, Austausch von SLICOT/RASP-Routinen.
- 2. Version von SLICOT 1993, ca. 90 aufrufbare Funktionen [22]. (Zum Vergleich: RASP hat z.Zt. ca. 320.)
- WGS und DLR vereinbaren wechselseitiges Kompatibilitätsrezept für SLICOT und RASP, um zukünftige Entwicklung zu rationalisieren [14].
- 1996/97: SLICOT wird von NAG losgelöst, NAG–Routinen werden durch BLAS– bzw. LAPACK–Routinen ersetzt.
- 3. Version Anfang 1998 (*public domain*) erhältlich unter

  `ftp://wgs.esat.kulenven.ac.be/pub/WGS/SLICOT/`

  `http://www.win.tue.nl/wgs/slicot`

[3] The Numerical Algorithms Group, Oxford, Großbritannien

SLICOT

(Für Details siehe [5, 31].)

- Organisation in Kapitel/Sektionen:
  - A: Analyse (der Eigenschaften dynamischer Systeme),
  - D: Datenanalyse,
  - F: Filterberechnung,
  - M: Mathematische Routinen,
  - S: Synthese,
  - T: Transformation (zwischen verschiedenen Modellrepräsentationen),
  - U: Hilfsmittel.
- Aufbau in zwei Ebenen:
  - aufrufbare, voll dokumentierte Routinen (z.Zt. ca. 100),
  - Hilfsroutinen.
- Auswahl der Algorithmen erfolgt nach
  - Nutzen,
  - Robustheit (*Numerische Stabilität*),
  - Effizienz (unter Berücksichtigung moderner Computerarchitekturen).
- Dokumentation Versionen 1 und 2 über NAG [21, 22], für 3. Version bisher nur online unter

  `ftp://wgs.esat.kuleuven.ac.be/pub/WGS/SLICOT/doc/`

  erhältlich.
- Rigorose Implementierungs- und Dokumentationsstandards [29, 31, 30].
- Sammlung von Benchmark-Beispielen zur Evaluation der Algorithmen:
  ↪ Validierung der berechneten Ergebnisse,
  ↪ Vergleich von Algorithmen,
  ↪ Effizienzbeurteilung,
  ↪ Verhalten in Extremsituationen,
  ↪ Test von neuen Algorithmen.

  Existiert z.Zt. für diskrete und kontinuierliche AREs [3], weitere sind in Vorbereitung [4].

Tests einiger SLICOT 3.0 Funktionen

Einige Routinen der neuesten Version von SLICOT wurden mit Hilfe des *NAGWare Gateway Generators* [23] in MATLAB eingebunden. Ein Vergleich mit äquivalenten MATLAB-Funktionen kann in [5] gefunden werden. Einige Beispiele:

- FFT[4]: Die SLICOT–Routine ist
  - 2–4 mal schneller,
  - bis zu zwei Größenordnungen genauer

  als die MATLAB–Routine `fft`.

- Kalman–Filter–Berechnung:
  Die Aufdatierung der Kovarianz-/Informationsmatrix (in faktorisierter Form) mit der SLICOT–Routine ist bei gleicher Genauigkeit 4–5 mal schneller als entsprechende Berechnungen in MATLAB.

- *Controllability Staircase Form*:
  Für Matrixpaar (A, B), $A \in \mathbb{R}^{n \times n}$, $B \in R^{n \times m}$ mit $n = 256$ erhält man:

Tabelle 2: CPU-Zeiten (Sun ULTRA 2 Workstation)

m	MATLAB (`ctrbf`)	SLICOT	Speed-Up
32	25.1 sec.	2.6 sec.	≈ 10
1	639.6 sec.	2.5 sec.	> 250

- *Pole Placement*:
  Häufige Antwort der MATLAB Control Toolbox Funktion `place`:

  ```
  Can't place eigenvalues there.
  ```

  SLICOT erzielt höhere Genauigkeit, und kann Pole legen, auch wenn `place` versagt. CPU Zeit (Sek.) für System der Ordnung $n = 128$:

  MATLAB 104.0 sec., SLICOT 0.55 sec. $\Longrightarrow$ Speed-Up Faktor $\approx$190.

[4] Fast Fourier Transformation

NICONET – Numerics In COntrol NETwork

- WGS-Initiative 1996/97 zum Start eines europäischen Netzwerks im **EU BRITE–EURAM III** Programm.
- Implementierungsphase 1998–2001.
- Budget ca. 1.5 Mio DM.
- 17 Partner in 7 Ländern (industrielle Partner in *italic*):

  D: ∗ Universität Bremen, Fachbereich 3 – Mathematik und Informatik, Zentrum für Technomathematik, (P. Benner);
  ∗ Technische Universität Chemnitz, Fakultät für Mathematik (V. Mehrmann);
  ∗ Deutsches Zentrum für Luft- und Raumfahrt e.V., Oberpfaffenhofen, Institut für Robotik und Systemdynamik (A. Varga);
  ∗ *TBZ-Pariv GmbH, Chemnitz (R. Wohlgemuth).*

  B: ∗ Katholieke Universiteit Leuven, ESAT — Departement Elektrotechniek (SISTA) (S. Van Huffel – Chairwoman);
  ∗ Université Catholique de Louvain, CESAME: Centre for Systems Engineering and Applied Mechanics (P. Van Dooren);
  ∗ *Leuven Measurement Systems International, Leuven (H. Van der Auwenaar).*

  NL: ∗ Technische Universiteit Eindhoven, Mathematik (A. Stoorvogel) und Dept. Elektrotechniek (A. van den Boom);
  ∗ Technische Universiteit Delft, Departement Elektrotechniek (M. Verhaegen);
  ∗ *Aspen Technology, Best (T. Backx).*

  GB: ∗ University of Leicester, Department of Engineering (D.-W. Gu, S. Spurgeon);
  ∗ *The Numerical Algorithms Group, Oxford (S. Hammarling).*

  F: ∗ Institut National de Recherche en Informatique et Automatique, Le Chesnay Cedex (F. Delebecque, S. Steer);
  ∗ *Société de Fabrication d'Intruments de Mesures, Massy Cedex (P. Coustal).*

  E: ∗ Universidad Politécnica de Valencia, Departamento de Sistemas Informáticos y Computación (V. Hernández);
  ∗ *Omron Electronics S.A., Madrid (F. Colás).*

  S: ∗ Umeå University, Department of Computing Science (B. Kågström).

(Siehe auch [6, 27].)

Inhalte des NICONETs

Task P: Vorbereitende Aufgaben:

a) Standard für parallele SLICOT Version.
b) Standard für Behandlung nichtlinearer Systeme.
c) SLICOT Benchmark-Sammlung.
d) Vorbereitung für Einsatz elektronische Medien
(Newsletter, WWW, e-mail, Online–Präsentationen, etc.).

Task I: Grundlegende numerische Software zur Ergänzung/Erweiterung von SLICOT.

Task II: Modellreduktion (Reduktion der Dimension des Modells/Reglers).

Task III: Systemidentifizierung (für lineare/nichtlinear Systeme).

Task IV: Entwurf robuster Regler:

a) LQ/H_2 und H_∞-Regler,
b) H_∞-Loop Shaping/μ–synthesis,
c) LMI–based design.

Task V: Simulation, Identifizierung, Regelung nichtlinearer Systeme in der Robotik.

Task S: "Public Relations":
a) SLICOT FTP-Server/Webpage,
b) NICONET Newsletter,
c) Präsentation bei Konferenzen/Workshops,
d) Organisation von Workshops, Kursen, Tutorien.

Literatur

Einige der unten angegebenen Referenzen sind als Postscript–Dateien von folgender WWW–Seite abrufbar:

`http://www.win.tue.nl/wgs/reports.html`

Informationen zur WGS findet man unter

`http://www.win.tue.nl/wgs/`

und zu NICONET unter

`http://www.win.tue.nl/wgs/niconet.html.`

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, second edition, 1994.

[2] W.F. Arnold, III and A.J. Laub. Generalized eigenproblem algorithms and software for algebraic Riccati equations. *Proc. IEEE*, 72:1746–1754, 1984.

[3] P. Benner, A. J. Laub, and V. Mehrmann. Benchmarks for the numerical solution of algebraic Riccati equations. *IEEE Control Systems Magazine*, 7(5):18–28, 1997.

[4] P. Benner and V. Mehrmann. Collections of benchmark examples in control. *WGS Newsletter*, 10:7–8, 1996.

[5] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, and A. Varga. SLICOT - a subroutine library in systems and control theory. *Applied and Computational Control, Signals, and Circuits*, 1, 1998. To appear. (See also: NICONET Report 97-3, June 1997.)

[6] A.J.W. van den Boom, S. Van Huffel, and P. Benner. NICONET, a Network for Numerically Reliable Software in CACSD. *Journal A – Benelux Quarterly Journal on Automatic Control*, 38(3):20–21, 1997.

[7] The Boeing Company, Seattle, WA. *EASY5 User's Guide*, 1996.

[8] F. Delebecque and S. Steer. *Integrated Scientific Computing with Scilab*. Birkhäuser, Boston, MA, 1997.

[9] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling. A set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Soft.*, 16:1–17, 1990.

[10] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson. An extended set of FORTRAN Basic Linear Algebra Subprograms. *ACM Trans. Math. Soft.*, 14:1–17, 1988.

[11] P. Gahinet, A. Laub, and A. Nemirovski. The LMI Control Toolbox. The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760, 1995.

[12] G. Grübel. Die regelungstechnische Programmbibliothek RASP. *Regelungstechnik*, 31:75–81, 1983.

[13] G. Grübel, H.-D. Joos, M. Otter, and R. Finsterwalder. The ANDECS design environment for control engineering. In *Prepr. of 12th IFAC World Congress, Sydney, Australia*, 1993.

[14] G. Grübel, A. Varga, A. J. W. van den Boom, and A. J. Geurts. Towards a coordinated development of numerical CACSD software: the RASP/SLICOT compatibility concept. In *Proc. CACSD'94 Symposium*, pages 499–504, Tucson, AZ, 1994.

[15] Integrated Systems, Inc., Santa Clara, CA. *Xmath Basics, Version 5.0*, 1997.

[16] C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh. Basic linear algebra subprograms for FORTRAN usage. *ACM Trans. Math. Software*, 5:303–323, 1979.

[17] The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, Mass, 01760. *The MATLAB Control Toolbox, Version 4.0*, 1997.

[18] The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, Mass, 01760. *The MATLAB Robust Control Toolbox, Version 2.0b*, 1994.

[19] The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, MA 01760. *Using* MATLAB*, Version 5.1*, 1997.

[20] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler. *Matrix Eigensystem Routines*—EISPACK *Guide*, volume 6 of *Lecture Notes in Computer Science*. Springer, New York, second edition, 1976.

[21] The Numerical Algorithms Group – NAG, Wilkinson House, Jordan Hill Road, Oxford, OX2 8DR, U.K. *NAG SLICOT Library Manual, Release 1*, 1991.

[22] The Numerical Algorithms Group – NAG, Wilkinson House, Jordan Hill Road, Oxford, OX2 8DR, U.K. *NAG SLICOT Library Manual, Release 2*, 1993. (Updates Release 1 of May 1990.)

[23] The Numerical Algorithms Group – NAG, Wilkinson House, Jordan Hill Road, Oxford OX2 8DR, UK. *NAGWare Gateway Generator, Release 2.0*, 1994.

[24] A. J. W. van den Boom, A. Brown, F. Dumortier, A. Geurts, S. Hammarling, R. Kool, M. Vanbegin, P. Van Dooren, and S. Van Huffel. SLICOT, a subroutine library in control and systems theory. In *Proc. of the 5th IFAC Symposium on Computer Aided Design in Control Systems, Swansea, UK*, pages 89–94, Oxford, UK, July 1991. Pergamon Press.

[25] A. J. W. van den Boom and S. Van Huffel. Developments around the freeware standard control library SLICOT. In *Proc. CACSD'96 Symposium*, pages 473–476, Dearborn, MI, 1996.

[26] P. Van Dooren. A generalized eigenvalue approach for solving Riccati equations. *SIAM J. Sci. Statist. Comput.*, 2:121–135, 1981.

[27] S. Van Huffel and A. J. W. van den Boom. NICONET: network for performant numerical software development in control engineering. In *Proc. 7th IFAC Symposium on Computer-Aided Control Systems Design*, pages 197–202, Ghent, Belgium, April 28–30 1997.

[28] The Working Group on Software – WGS. *An inventory of basic software for computer aided control system design*, 1985. WGS-report 85-1.

[29] The Working Group on Software – WGS. *SLICOT Implementation and Documentation Standards*, 1990. WGS-report 90-1.

[30] The Working Group on Software – WGS. *SLICOT Contributor's Kit 2.1*, 1994. WGS-report 96-2.

[31] The Working Group on Software – WGS. *SLICOT Implementation and Documentation Standards 2.1*, 1996. WGS-report 96-1.

[32] The Working Group on Software – WGS. *Results of the NICONET Questionnaire*, 1997. WGS-report 97-1.