GELDA — Ein Softwarepaket zur Lösung linearer differentiell–algebraischer Gleichungen mit beliebigem Index

Peter Kunkel[1], Volker Mehrmann[2], Werner Rath[2] und Jörg Weickert[2]

[1] Fachbereich Mathematik, Carl von Ossietzky Universität, D-26111 Oldenburg
[2] Fakultät für Mathematik, TU Chemnitz–Zwickau, D-09107 Chemnitz

Summary: GELDA ist ein neues FORTRAN 77 Softwarepaket zur numerischen Lösung linearer differentiell–algebraischer Gleichungen mit variablen Koeffizienten. Die Implementierung basiert auf einem neuen Diskretisierungsverfahren und kann Systeme mit beliebigem Index lösen. Außerdem können Systeme mit nichteindeutiger Lösung oder inkonsistenten Anfangswerten gelöst werden.
Nach einem kurzen Überblick über frei verfügbare Software zur numerischen Lösung von differentiell–algebraischen Gleichungen gehen wir auf einige Punkte zur Implementierung von GELDA ein.

1 Einleitung

Bei der mathematischen Modellierung vieler physikalischer Vorgänge treten Differentialgleichungssysteme auf. Oft kommen neben den Differentialgleichungen auch algebraische Gleichungen vor, die Zwangsbedingungen (etwa die Einschränkung der Bewegung eines Massepunktes in der Ebene) oder Erhaltungsgleichungen (etwa die Kirchhoffschen Gesetze bei elektrischen Schaltungen) beschreiben. Solche gemischten Systeme nennt man daher differentiell–algebraische Gleichungssysteme. Die allgemeinste Form differentiell–algebraischer Gleichungen ist gegeben durch

$$F(t, x(t), \dot{x}(t)) = 0, \qquad t \in [t_1, t_2], \tag{1.1}$$

wobei $\dot{x}$ die Ableitung von x nach t bezeichnet, eventuell zusammen mit einer Anfangsbedingung

$$x(t_0) = x_0, \tag{1.2}$$

wobei $t_0 \in [t_1, t_2]$ und $x_0 \in \mathcal{C}^n$. Die Funktion F ist dabei gegeben und gesucht wird eine Funktion x, die die Gleichung (1.1) löst und eventuell (1.2) erfüllt.

Ein für die Anwendungen wichtiger Spezialfall der allgemeinen differentiell–algebraischen Gleichungen (1.1) sind linear implizite Systeme der Form

$$B(t, x(t))\dot{x}(t) = f(t, x(t)), \qquad t \in [t_1, t_2], \tag{1.3}$$

wobei $B(t, x)$ eine $n \times n$ Matrix ist, deren Einträge von t und x abhängen. Verwendet man zur Modellierung eines mechanischen Mehrkörpersystems den Euler–Lagrange–Ansatz, so erhält man z.B. eine differentiell–algebraische Gleichung der Form (1.3).

Ein weiterer Spezialfall der allgemeinen differentiell–algebraischen Gleichungen (1.1) sind lineare Gleichungen mit variablen Koeffizienten, gegeben durch

$$E(t)\dot{x}(t) = A(t)x(t) + f(t), \qquad t \in [t_1, t_2]. \tag{1.4}$$

Dabei sind E und A quadratische Matrizen, deren Einträge Funktionen von t sind.

Probleme dieses Typs erhält man unter anderem durch Linearisierung von (1.1) oder (1.3) entlang einer Trajektorie. Mechanische Mehrkörpersysteme haben häufig die Form (1.4), da bei vielen dieser Systeme in (1.3) B unabhängig von x und f linear in x ist.

Die wichtigste Invariante für die Analyse von differentiell–algebraischen Gleichungen ist der sogenannte Index. In der Literatur finden sich verschiedene Indexbegriffe, z.B. der Differentiations–Index [2], der Strangeness–Index [9], der Störungs–Index [7], usw.

Am häufigsten wird der Differentiations–Index (d–Index) benutzt, da er eine große Klasse von praxisrelevanten Problemen abdeckt und in vielen Fällen relativ einfach zu berechnen ist.

Im wesentlichen ist der d–Index die Zahl, die man an Ableitungen der algebraischen Gleichungen benötigt, um das differentiell–algebraische Gleichungssystem in ein System gewöhnlicher Differentialgleichungen zu überführen. So hat das System

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) + f_1(t) \\ 0 &= x_2(t) + f_2(t) \end{aligned}$$

den d-Index 1, da man die zweite Gleichung einmal ableiten muß, um ein System von zwei gewöhnlichen Differentialgleichungen zu erhalten.

Für differentiell–algebraische Gleichungen der Form (1.4) wurde in [9] der Strangeness–Index (s–Index) eingeführt. Der s–Index verallgemeinert den d–Index für rechteckige Systeme, d.h. E und A sind rechteckige Matrizen. Insbesondere ist der s-Index auch für Systeme mit unbestimmten Lösungskomponenten definiert, wie sie z.B. bei der Lösung von linear quadratischen Optimalsteuerungsproblemen und differentiell–algebraischen Riccatigleichungen auftreten.

Falls für eine differentiell–algebraische Gleichung (1.4) der d–Index ν existiert, so ist auch der s–Index μ definiert und es gilt

$$\mu = \max\{0, \nu - 1\}. \tag{1.5}$$

Im nächsten Abschnitt geben wir einen kurzen Überblick über numerische Software zur Lösung differentiell–algebraischer Gleichungen. Anschließend gehen wir noch auf einige Punkte der Implementierung von GELDA ein.

2 Software für differentiell–algebraische Gleichungen

Seit Anfang der 80er Jahre ist Software zur Lösung von differentiell–algebraischen Gleichungen frei verfügbar. Traditionell handelt es sich dabei um FORTRAN Routinen. Einer der ersten Codes war DASSL von L. Petzold [2]. DASSL gehört zur Familie der Mehrschrittverfahren und ist zur Lösung von d–Index–1–Systemen der Form (1.1) konstruiert worden.

Während Mehrschrittverfahren zur Lösung von gewöhnlichen Differentialgleichungen relativ einfach auf d–Index–1–Systeme der Form (1.1) übertragen werden können, ist dies für Extrapolations- und Einschrittverfahren wesentlich schwieriger. Die meisten zur Zeit frei verfügbaren Codes dieser Gattungen sind zur Lösung der einfacher strukturierten linear impliziten Systeme der Form (1.3) konzipiert. Mit LIMEX von Deuflhard/Hairer/Zugck [4] liegt ein Extrapolationscode für linear implizite d-Index–1–Systeme vor. Ist die Matrix B in (1.3) unabhängig von t und x, so kann als Einschrittverfahren z.B. der Runge–Kutta–Code RADAU5 von Hairer/Wanner [7] angewendet werden. Im Gegensatz zu DASSL und LIMEX kann RADAU5 unter gewissen Voraussetzungen Systeme bis zum d-Index 3 lösen.

Die meisten der Standardlöser sind nur für d–Index–1–Systeme konzipiert. Bei der Modellierung von mechanischen Mehrkörpersystemen mit dem Euler–Lagrange–Ansatz treten differentiell–algebraische Gleichungen mit d–Index 3 auf. Speziell für diese Problemklasse wurden Löser wie MEXX von Lubich [12] und ODASSL von Führer [6] entwickelt, die die besondere Struktur dieser Systeme ausnutzen.

In jüngster Zeit geht man dazu über, elektronische Bauteile oder elastische Verbindungen in die Modellierung von Mehrkörpersystemen zu integrieren. Dabei kann allerdings die Struktur der Gleichungen zerstört werden und Codes wie MEXX oder ODASSL sind nicht mehr anwendbar.

Im Gegensatz zu den obigen Standardlösern ist GELDA zur Lösung von linearen differentiell–algebraischen Gleichungen der Form (1.4) von *beliebigem* s-Index konstruiert und es können Systeme mit *unbestimmten* Lösungskomponenten gelöst werden. Außerdem bereitet das häufig auftretende Problem von inkonsistenten Anfangswerten keine Schwierigkeiten, da sich konsistente Anfangswerte einfach berechnen lassen.

3 Implementierung von GELDA

Die Implementierung von GELDA beruht auf dem in [8] vorgestellten Diskretisierungsverfahren. Ausgangspunkt ist die Konstruktion eines numerischen Verfahrens, das die Bestimmung sogenannter charakteristischer Größen und damit auch des s–Index μ der differentiell–algebraischen Gleichung ermöglicht. Nach einer Idee von Campbell (siehe [3]) leiten wir dazu die Gleichung (1.4) sukzessive ab und bringen die Ableitungen von x auf die linke Seite. Faßt man die erhaltenen Gleichungen zu einem neuen großen differentiell–algebraischen Gleichungssystem zusammen, so erhält man für hinreichend glatte Funktionen E, A und f für $\ell = 1, \ldots, \mu$ Gleichungen der Form

$$M_\ell(t)\dot{z}_\ell(t) = N_\ell(t)z_\ell(t) + g_\ell(t). \tag{3.6}$$

Für festes $t \in [t_1, t_2]$ können nun lokale charakteristische Größen von (3.6) für $\ell = 1, \ldots, \mu$ berechnet werden. Entscheidend ist, daß diese lokalen Größen selbst wieder charakteristisch für die ursprüngliche differentiell–algebraische Gleichung (1.4) sind, und damit können die charakteristischen Größen von (1.4) samt s–Index μ aus den lokalen Größen von (3.6) berechnet werden. Außerdem erkennt man, daß für Systeme von höherem s–Index die Lösung von Informationen über die Ableitungen von E, A und f abhängt.

Im Gegensatz zu [3] wird aber nicht dieses große System gelöst. Stattdessen wird aus (3.6) mittels orthogonaler Projektionen von links eine strangeness–freie differentiell-algebraische Gleichung

$$\hat{E}(t)\dot{x}(t) = \hat{A}(t)x(t) + \hat{f}(t) \tag{3.7}$$

extrahiert. Die so gewonnene differentiell-algebraische Gleichung hat die gleiche Größe wie die ursprüngliche Gleichung (1.4), und da die gesuchte Funktion x nicht transformiert wurde, stimmen die Lösungsmengen beider Systeme überein.

Eine der grundlegenden Aufgaben bei dem hier beschriebenen Ansatz ist die Berechnung der charakteristischen Größen inklusive des s–Index und die Extraktion der strangeness–freien differentiell–algebraischen Gleichung (3.7). Um einen numerisch stabilen Algorithmus zu erhalten, verwenden wir nur orthogonale Transformationen. Zur Bestimmung des s–Index sind von (3.6) für $\ell = 0, \ldots, \mu$ jeweils drei aufeinanderfolgende Rangbestimmungen durchzuführen. Diese Rangbestimmungen können etwa mittels Singulärwertzerlegung oder QR-Zerlegung mit Spaltenaustausch bewerkstelligt werden. Um die strangeness–freie Gleichung (3.7) zu extrahieren, kann man diese Zerlegungen ebenfalls benutzen. Aus Gründen der numerischen Stabilität verwenden wir in der aktuellen Version von GELDA die Singulärwertzerlegung.

Bei der Implementierung von GELDA wurde besonderer Wert auf die Benutzung von Standardbibliotheken gelegt. Im Laufe der letzten Jahre hat sich das Programmpaket LAPACK [1] zur Lösung von Aufgaben der numerischen linearen Algebra durchgesetzt. LAPACK selbst baut auf den *basic linear algebra subroutines* (BLAS) auf, die die elementaren Operationen der numerischen linearen Algebra zur Verfügung stellen (siehe [11, 5]). Diese Operationen sind im wesentlichen die elementaren Vektor/Vektor–, Vektor/Matrix– und Matrix/Matrix–Operationen wie Addition und Multiplikation. Wir benutzen BLAS und LAPACK für alle Aufgaben der numerischen linearen Algebra, dazu zählen insbesondere die Berechnung der Singulärwertzerlegung und das Lösen von linearen Gleichungssystemen und Kleinste–Quadrate–Problemen.

Die zweite grundlegende Aufgabe in GELDA ist die numerische Lösung der strangeness–freien differentiell–algebraischen Gleichung (3.7). Die punktweise Berechnung der orthogonalen Projektionen liefert (3.7) allerdings nur bis auf eine eventuell nichtstetige, punktweise unitäre Skalierung von links. Glücklicherweise sind aber BDF- und Runge–Kutta Verfahren invariant gegen solche Transformationen der differentiell–algebraischen Gleichung (3.7). In GELDA verwenden wir den BDF Code DASSL und den Runge-Kutta Code RADAU5 zur Lösung der strangeness–freien Gleichung. Beide Verfahren sind so modifiziert worden, daß sie die Struktur der Gleichung (3.7) ausnutzen und in jedem Punkt, in dem diese Gleichung benötigt wird, den Reduktionsprozeß aufrufen. Weiterhin wurden beide Verfahren auf die Verwendung von BLAS und LAPACK umgestellt.

Neben der Verwendung von Standardbibliotheken ist die ausführliche Dokumentation des Verfahrens ein wichtiger Gesichtspunkt. Als Implementations- und Dokumentationsstandard wurde der SLICOT Standard gewählt (siehe [13]). Dieser Standard legt genau fest, wie die Prozedurschnittstelle aussieht und die zugehörige Dokumentation aufgebaut ist. Zur Dokumentation der vom Benutzer aufrufbaren Routinen gehören sowohl die genaue Beschreibung der Prozedurargumente, wie auch Beschreibungen der Fehlermeldungen, die eine Prozedur zurückliefert, eine Beschreibung des verwendeten Algorithmus, eine Liste von Referenzen und Informationen über numerische Aspekte der Implementierung. Diese Informationen sind sowohl im Quelltext enthalten, als auch in gedruckter Form verfügbar. Die gedruckte Dokumentation enthält außerdem ein komplettes Beispielprogramm mit Eingabe- und Ausgabedaten.

GELDA und die zugehörige Dokumentation sind frei verfügbar und können bei den Autoren angefordert oder über NETLIB[1] bezogen werden. Zum Programmpaket gehören die FORTRAN Quellcodes mit Inline Dokumentation und die LaTeX Quellen der Dokumentation, die zu vom Benutzer aufrufbaren Routinen gehören. Neben diesen Prozedurbeschreibungen ist auch eine Arbeit [10] erhältlich, die ausführlicher auf die Implementierung und Anwendung von GELDA

[1] NETLIB ist unter anderem über die URL http://www.netlib.org erreichbar. GELDA findet man dort unter der URL http://www.netlib.org/ode/dgelda.tar.gz

eingeht.

4 Zusammenfassung

GELDA ist ein neues FORTRAN 77 Programmpaket zur numerischen Lösung von linearen differentiell-algebraischen Gleichungen mit variablen Koeffizienten. Im Gegensatz zu den Standardlösern kann GELDA Probleme mit beliebigem Index, unbestimmten Lösungskomponenten und inkonsistenten Anfangsbedingungen lösen.

Bei der Implementierung von GELDA wurde besonderer Wert auf die Verwendung von Standardbibliotheken und eine ausführliche Dokumentation gelegt. Das komplette Programmpaket ist inklusive Dokumentation frei verfügbar.

Neben der Pflege des Programmpaketes GELDA ist die Erweiterung von GELDA und der zugrundeliegenden Theorie für nichtlineare differentiell–algebraische Gleichungen und lineare Steuerungsprobleme aktueller Forschungsgegenstand.

Literaturverzeichnis

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammerling, A. McKenney, S. Ostrouchov, and D. Sorenson. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, 2nd edition, 1995.

[2] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial-Value Problems in Differential Algebraic Equations*, volume 14 of *Classics in Applied Mathematics*. SIAM, Philadelphia, PA, 1996.

[3] S. L. Campbell. A general form for solvable linear time varying singular systems of differential equations. *SIAM J. Math. Anal.*, 18(4):1101–1115, July 1987.

[4] P. Deuflhard, E. Hairer, and J. Zugck. One step and extrapolation methods for differential-algebraic systems. *Numer. Math.*, 51:501–516, 1987.

[5] J. J. Dongarra, J. Du Croz, S. Hammerling, and R. J. Hanson. Algorithm 656: An extended set of FORTRAN basic linear algebra subprograms. *ACMTM*, 14:18–32, 1988.

[6] C. Führer. *Differential-Algebraische Gleichungsysteme in Mechanischen Mehrkörpersystemen*. PhD thesis, Math. Institute, Technische Universität, München, 1988.

[7] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II*. Springer-Verlag, Berlin, 1991.

[8] P. Kunkel and V. Mehrmann. A new class of discretization methods for the solution of linear differential-algebraic equations. Materialien LXII , FSP Mathematisierung, Universität Bielefeld, 1992. To appear in SIAM J. Numer. Anal.

[9] P. Kunkel and V. Mehrmann. Canonical forms for linear differential-algebraic equations with variable coefficients. *J. Comput. Appl. Math.*, 56:225–259, 1994.

[10] P. Kunkel, V. Mehrmann, W. Rath, and J. Weickert. GELDA: A software package for the solution of general linear differential algebraic equations. Preprint SPC 95_8, TU Chemnitz-Zwickau, February 1995.

[11] C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh. Basic linear algebra subprograms for FORTRAN usage. *ACM Trans. Math. Software*, 5:308–323, 1979.

[12] C. Lubich, U. Nowak, U. Pöle, and C. Engstler. MEXX–Numerical software for the integration of constrained mechanical multibody systems. Technical Report SC 92-12, ZIB, Berlin, 1992.

[13] Working Group on Software. SLICOT implementation and documentation standards. WGS-Report 90-1, Eindhoven, Oxford, 1990.