Otto-von-Guericke-Universität Magdeburg
Max-Planck-Institut für Dynamik komplexer technischer Systeme
Computational Methods for Systems and Control Theory
Martin Stoll, Tobias Breiten, Jessica Bosch

Sommersemester 2013

Linear Systems and Matrix Equations – Bartels Stewart Algorithm.

Suchen die Lösung der Lyapunovgleichung

$$AX + XA^T = W \tag{1}$$

mit $A \in \mathbb{R}^{n\times n}$, sowie $W \in \mathbb{R}^{n\times n}$ positiv definit und symmetrisch. Die Lösung des zur Lyapunovgleichung äquivalenten linearen Gleichungssystems

$$((A \otimes I_n) + (I_n \otimes A))\, vec\, X = vec\, W \tag{2}$$

mit dem Gaußschen Elimininationsverfahren würde $\frac{2}{3}(n^2)^3 = \frac{2}{3}n^6$ Flops kosten und auf einen Speicherbedarf für n^4 reelle Zahlen hinauslaufen. Dies ist bereits für $n = 100$ nicht mehr vertretbar. (Es würden bereits mehr als 800MB Hauptspeicher benötigt!) Unser Ziel ist es, einen Algorithmus mit Rechenaufwand $\mathcal{O}(n^3)$ und Speicherbedarf $\mathcal{O}(n^2)$ zu entwickeln. Dies gelingt, da (2) ein Gleichungssystem mit sehr spezieller Struktur in $\mathbb{R}^{n^2\times n^2}$ ist und durch $\mathcal{O}(n^2)$ Daten definiert ist.
Die Idee ist, die Lyapunovgleichung (1) in eine Gestalt zu transformieren, so daß das Gleichungssystem (2) durch einfaches Rückwärtseinsetzen mit einem Aufwand von $\mathcal{O}(n^3)$ gelöst werden kann.
Dazu verwenden wir ein Ergebnis aus der numerischen linearen Algebra.

Theorem 1 (Reelle Schur-Zerlegung)**.** *Sei $A \in \mathbb{R}^{n\times n}$. Dann existiert $Q \in \mathbb{R}^{n\times n}$ orthogonal, so daß*

$$A = QTQ^T, \quad T = \begin{bmatrix} T_{11} & \dots & T_{1,r} \\ & \ddots & \vdots \\ & & T_{rr} \end{bmatrix} \text{ mit } T_{jj} \in \mathbb{R}^{n_j\times n_j}, n_j \in \{1,2\}, \quad \sum_{j=1}^{r} n_j = n, \tag{3}$$

wobei für $n_j = 1$, T_{jj} ein reeller Eigenwert von A ist und für $n_j = 2$ sind die Eigenwerte von T_{jj} ein komplex-konjugiertes Eigenwertpaar von A.
Außerdem kann Q so gewählt werden, daß die Eigenwerte von A in beliebiger Reihenfolge auf der Diagonale von T erscheinen.

Beweis. Siehe z.B. [1, 2]. □

Die Zerlegung von A in (3) heißt *Schur-Zerlegung* von A, T heißt dann *Schurform* von A und die Spalten von Q sind die *Schurvektoren* von A. Beachte, daß die ersten $n_1 + \ldots + n_k$ Spalten von Q einen A-invarianten Unterraum bilden für alle $k = 1, \ldots, r$. Die Schur-Zerlegung von A kann z.B. mit dem QR Algorithmus numerisch rückwärts stabil berechnet werden, siehe z.B. [1, 2].
Multipliziert man die Lyapunovgleichung (1) von rechts mit der Schurvektor-Matrix Q von A und von links mit Q^T, so ergibt sich die äquivalente Lyapunovgleichung

$$\begin{bmatrix} A_1 & A_2 \\ 0 & A_3 \end{bmatrix}\begin{bmatrix} X_1 & X_2 \\ X_2^T & X_3 \end{bmatrix} + \begin{bmatrix} X_1 & X_2 \\ X_2^T & X_3 \end{bmatrix}\begin{bmatrix} A_1^T & 0 \\ A_2^T & A_3^T \end{bmatrix} = \begin{bmatrix} W_1 & W_2 \\ W_2^T & W_3 \end{bmatrix}$$

mit $A_3 \in \mathbb{R}^{n_r\times n_r}$. Unter Ausnutzung der Symmetrie zerfällt diese Lyapunovgleichung in die folgenden drei linearen Matrixgleichungen:

$$\begin{aligned} A_1X_1 + X_1A_1^T &= W_1 - A_2X_2^T - X_2A_2^T =: \tilde{W}_1 = \tilde{W}_1^T, && (4)\\ A_1X_2 + X_2A_3^T &= W_2 - A_2X_3 =: \tilde{W}_2, && (5)\\ A_3X_3 + X_3A_3^T &= W_3. && (6) \end{aligned}$$

Die Gleichung (6) kann nun explizit aufgelöst werden. Falls $n_r = 1$, dann ist (6) skalar und

$$X_3 = \frac{W_3}{2A_3},$$

da $A_3 \neq 0$ (andernfalls wäre die Lyapunovgleichung nicht eindeutig lösbar). Im Fall $n_r = 2$ verwendet man explizit die vektorisierte Darstellung von (6). Unter Ausnutzung der Symmetrien ergibt sich das Gleichungssystem

$$\begin{bmatrix} a_{n-1,n-1} & a_{n-1,n} & 0 \\ a_{n,n-1} & a_{n-1,n-1} + a_{n,n} & a_{n-1,n} \\ 0 & a_{n,n-1} & a_{n,n} \end{bmatrix} \begin{bmatrix} x_{n-1,n-1} \\ x_{n-1,n} \\ x_{n,n} \end{bmatrix} = \begin{bmatrix} w_{n-1,n-1}/2 \\ w_{n-1,n} \\ w_{n,n}/2 \end{bmatrix}. \tag{7}$$

Die Lösung von (7) sollte mit einer LU Zerlegung mit vollständiger Pivotisierung erfolgen, um möglichst hohe Genauigkeit zu erzielen.
Die Lösung X_3 wird nun in (5) eingesetzt. Damit ergibt sich eine Sylvestergleichung, die eindeutig lösbar ist (wegen Satz 2.4 und $\Lambda(A_1) \cap \Lambda(-A_3^T) = \emptyset$, falls $\Lambda(A) \cap \Lambda(-A^T) = \emptyset$). Aufgrund der sehr speziellen Struktur dieser Sylvestergleichung kann man diese Lösung einfach durch Rückwärtseinsetzen berechnen. Partitioniere dazu analog zu (3),

$$A_1 = \begin{bmatrix} A_{11} & \dots & A_{1,r-1} \\ & \ddots & \vdots \\ & & A_{r-1,r-1} \end{bmatrix}, \quad X_2 = \begin{bmatrix} x_1 \\ \vdots \\ x_{r-1} \end{bmatrix}, \quad \tilde{W}_2 = \begin{bmatrix} w_1 \\ \vdots \\ w_{r-1} \end{bmatrix}, \quad x_j, w_j \in \mathbb{R}^{n_j \times n_r}.$$

Dann kann man die x_j rückwärts berechnen aus

$$A_{jj}x_j + x_j A_3^T = w_j - \sum_{i=j+1}^{r-1} A_{j,i}x_i =: \tilde{w}_j, \qquad j = r-1, r-2, \dots, 1.$$

Bei der Lösung dieser Gleichung muß man vier Fälle unterscheiden.

$n_j = 1, n_r = 1$**:** Die Gleichung ist skalar und $x_j = \frac{\tilde{w}_j}{A_{jj}+A_3}$.

$n_j = 2, n_r = 1$**:** Man erhält ein Gleichungssystem im $\mathbb{R}^2$ mit eindeutiger Lösung,

$$(A_{jj} + A_3 \cdot I_2)x_j = \tilde{w}_j.$$

$n_j = 1, n_r = 2$**:** Man erhält ein Gleichungssystem im $\mathbb{R}^2$ mit eindeutiger Lösung,

$$(A_{jj} \cdot I_2 + A_3)x_j^T = \tilde{w}_j^T.$$

$n_j = 2, n_r = 2$**:** Man erhält ein Gleichungssystem im $\mathbb{R}^4$ mit eindeutiger Lösung,

$$\left((I_2 \otimes A_{jj}) + (A_3^T \otimes I_2)\right) vec\, x_j = vec\, \tilde{w}_j.$$

Die Gleichungssysteme in den letzten drei Fällen werden mit LU Zerlegung und vollständiger Pivotisierung gelöst.
Setzt man nun die so berechnete Lösung von (5) in (4), so erhält man eine Lyapunovgleichung in $\mathbb{R}^{n-n_r \times n-n_r}$ mit Koeffizientenmatrix in Schurform. Damit läßt sich diese Gleichung wieder aufteilen wie in (4)–(6), so daß sich das oben beschriebene Vorgehen rekursiv anwenden läßt bis (4) eine Gleichung im $\mathbb{R}^{n_1 \times n_1}$ ist, die direkt aufgelöst werden kann.
Zum Schluß muß man noch die berechnete Lösung in das ursprüngliche Koordinatensystem zurücktransformieren. Damit erhält man insgesamt den Algorithmus 1.

Algorithmus 1 Bartels-Stewart Algorithmus (1972)

Input: $A, W \in \mathbb{R}^{n\times n}$, $W = W^T$.

Output: Lösung $X = X^T$ der Lyapunovgleichung (1).

1: Berechne die reelle Schurform von A wie in (3) mit Hilfe des QR Algorithmus.
2: **if** $\Lambda(A) \cap \Lambda(-A) \neq \emptyset$ **then**
3: STOP; keine eindeutige Lösung.
4: **end if**
5: $\begin{bmatrix} W_1 & W_2 \\ W_2^T & W_3 \end{bmatrix} := Q^T W Q$, $W_3 \in \mathbb{R}^{n_r \times n_r}$.
6: $k := r$
7: **while** $k > 1$ **do**
8: Löse (6) mit $A_3 = A_{kk}$.
9: Löse (5) mit $A_1 = \begin{bmatrix} A_{11} & \dots & A_{1,k-1} \\ & \ddots & \vdots \\ & & A_{k-1,k-1} \end{bmatrix}$.
10: $W_1 := W_1 - A_2 X_2^T - X_2 A_2^T$.
11: $k := k - 1$
12: **end while**
13: Löse (4) als lineares Gleichungssystem (unter Ausnutzung der Symmetrie) im $\mathbb{R}^3$.
14: $X := Q\tilde{X}Q^T$

Der Bartels-Stewart Algorithmus benötigt ca. $32n^3$ elementare Rechenoperationen, dabei entfallen auf den QR Algorithmus ca. $25n^3$ Flops und auf die Schritte 5 und 14 je $3n^3$. Der gesamte Prozeß des Rückwärtseinsetzens (die **while**-Schleife) benötigt nur n^3 Operationen. Da sowohl der QR Algorithmus als auch das Rückwärtseinsetzen numerisch rückwärts stabil sind und darüberhinaus nur orthogonale Ähnlichkeitstransformationen verwendet werden, kann der Bartels-Stewart Algorithmus als numerisch rückwärts stabil angesehen werden.
Der Bartels-Stewart Algorithmus zur Lösung von Lyapunovgleichungen ist z.B. in der MATLAB Control Toolbox Funktion `lyap` implementiert.

Literatur

[1] G.H. GOLUB AND C.F. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, third ed., 1996.

[2] J. STOER AND R. BULIRSCH, *Einführung in die Numerische Mathematik II*, Springer-Verlag, Berlin, 3. ed., 1990. In German.