A PERIODIC SYSTEMS TOOLBOX FOR MATLAB

A. Varga

German Aerospace Center, DLR - Oberpfaffenhofen
Institute of Robotics and Mechatronics
D-82234 Wessling, Germany
Andras.Varga@dlr.de

Abstract: The recently developed PERIODIC SYSTEMS Toolbox for MATLAB is described. The basic approach to develop this toolbox was to exploit the powerful object manipulation features of MATLAB via flexible and functionally rich high level *m*-functions, while simultaneously enforcing highly efficient and numerically sound computations via the *mex*-function technology of MATLAB to solve critical numerical problems. The *m*-functions based user interfaces ensure user-friendliness in operating with the functions of this toolbox via an object oriented approach to handle periodic system descriptions. The *mex*-functions are based on FORTRAN implementations of recently developed structure exploiting and structure preserving numerical algorithms for periodic systems which completely avoid forming of lifted representations.

Keywords: Periodic systems, multi-rate systems, linear systems, numerical methods, computer-aided control systems design.

1. WHY A PERIODIC SYSTEMS TOOLBOX?

Many control applications are formulated as genuine periodic control problems as for example, satellite attitude control, helicopter forward flight control, orbital stabilization of underactuated systems, etc. Besides that, periodic systems represent a general framework to analyze and design multi-rate sampled-data systems. Solving robust control applications can also benefit of the increased stabilization potential of periodic control laws, as for example when solving simultaneous multi-model output feedback stabilization problems. This need to solve challenging periodic control applications led to a sustained effort in the last years to develop efficient and numerically reliable algorithms for periodic systems which can serve as basis for robust numerical software implementations. Developing object-oriented manipulation based tools relying on the newly developed algorithms fulfills in this way the increasing need for an user-friendly *computer-aided control systems design* (CACSD) environment dedicated to periodic systems.

The PERIODIC SYSTEMS Toolbox is primarily intended to extend the functionality of the CONTROL Toolbox of MATLAB by allowing the manipulation of the special class of periodic time-varying linear systems. Many functions can be seen to be formally equivalent to similar ones for standard systems which are available in the CONTROL Toolbox. The new Toolbox relies on many new numerically stable algorithms which fully exploit the underlying problem structure (so-called *fast* algorithms) or even preserve the problem structure (so-called *strongly stable* algorithms). This Toolbox is the first one providing comprehensive facilities for a user-friendly operation with periodic systems. Most of computational functions allow a convenient user-friendly operation to solve challenging periodic systems analysis problems, as for example, problems with large (even time-varying) state dimensions or with large periods.

The algorithms for periodic systems belong to three basic categories. The *lifting-based approaches* in the first category either involve forming explicitly matrix products or lead to very large dimensional standard problems. Therefore, these approaches are not suitable for reliable and, respectively, efficient numerical computations. In the second category, there are the so-called "fast" algorithms representing *structure exploiting numerically stable methods*. These methods are highly efficient, because completely avoid explicitly forming of lifted representations. Typical algorithms in this category are the algorithm to compute zeros of periodic systems or for solving periodic Riccati equations. The structurally stable algorithms in the third category are *structure preserving numerically stable methods*. Most algorithms in this category are very recent developments. Besides the traditional computational ingredients like the algorithms to compute periodic Hessenberg and Schur form (Bojanczyk *et al.*, 1992; Hench and Laub, 1994), algorithms to compute periodic Kronecker-like forms have been developed (Varga, 2004*c*). These algorithms as well as particularization of them have immediate applications in computing periodic systems poles and zeros, minimal realizations, solution of periodic Lyapunov and Riccati equations. Recently developed reliable algorithms to compute left/right inverses of periodic systems (Varga, 2004*a*) or left/right annihilators (Varga, 2004*e*) also rely on the computation of periodic Kronecker-like forms.

The requirements for satisfactory algorithms for periodic systems have been formulated in (Varga and Dooren, 2001). Besides generality (e.g., covering also problems with time-varying dimensions), a main requirement is the *backward stability* in terms of original system matrices. Structure preserving algorithms fulfilling this requirement are also called *structurally backward stable*, because they compute results which are exact for slightly perturbed original system data. The main advantage of such algorithms is that they produce guaranteed accurate results for well-conditioned problems. The weaker notion of backward stability achieved by structure exploiting (but *without* structure preserving) is for many computations still acceptable, since for many problems no structurally stable algorithms are presently available. The key ingredients to promote numerical stability are using exclusively orthogonal transformations, avoiding completely forming of products of non-orthogonal matrices, and fully exploiting structure by employing condensed forms.

A third main requirement is a high computational efficiency. Satisfactory algorithms from this point of view avoid excessive storage usage and have a computational complexity of at most $O(Nn^3)$, where n is the maximal state/input/output vector dimension and N is the discrete-time system period. It is a highly appealing feature that, according to above requirements, most of algorithms employed in the PERIODIC SYSTEMS Toolbox are completely satisfactory.

2. OBJECT-ORIENTED MANIPULATION

2.1 Periodic system representations

The toolbox can handle continuous-time periodic systems of the form

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) + B(t)u(t) \\ y(t) &= C(t)x(t) + D(t)u(t) \end{aligned} \tag{1}$$

where $A(t) \in \mathbb{R}^{n\times n}$, $B(t) \in \mathbb{R}^{n\times m}$, $C(t) \in \mathbb{R}^{p\times n}$, and $D(t) \in \mathbb{R}^{p\times m}$ are periodic matrices of period T.

Similarly, the toolbox can handle discrete-time periodic systems of the form

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k \\ y_k &= C_k x_k + D_k u_k \end{aligned} \tag{2}$$

where $A_k \in \mathbb{R}^{n_{k+1}\times n_k}$, $B_k \in \mathbb{R}^{n_{k+1}\times m_k}$, $C_k \in \mathbb{R}^{p_k\times n_k}$, and $D_k \in \mathbb{R}^{p_k\times m_k}$ are N-periodic matrices ($N \geq 1$). N is also called the discrete period.

To specify a system of the form (1), the system matrices can be entered in one of the three supported forms:

1. **symbolic matrix**: for example,

$$A(t) = \begin{bmatrix} \cos t & 1 \\ 1 & 1-\sin t \end{bmatrix}$$

   can be entered as a symbolic matrix (Extended Symbolic Toolbox required)

```
t = sym('t');
a = [cos(t) 1; 1 1-sin(t)]
```

2. **harmonic representation**: to represent

$$A(t) = A_0 + \sum_{i=1}^{n_h} \left(A_{ic} \cos t\frac{2\pi}{T} + A_{is} \sin t\frac{2\pi}{T} \right) \tag{3}$$

   a 3-dimensional complex array can be used to store A_0 and $A_{ic} + jA_{is}$ for $i = 1, \ldots, n_h$. For example,

$$B(t) = \begin{bmatrix} \cos t + \sin t \\ 1 - \sin t \end{bmatrix}$$

   can be entered as follows

```
b(:,:,1) = [  0;  1]; % B_0
b(:,:,2) = [ 1+j; -j]; % B_1c+j*B_1s
```

3. **function handle**: for example, to represent $C(t) = [\sin t + \cos 2t \;\; 1]$, the following *m*-function `c.m` can be defined

```
function ct = c(t)
ct = [ sin(t)+cos(2*t) 1];
```

A continuous-time periodic system with the matrices $A(t)$, $B(t)$, $C(t)$ above and $D(t) = 0$ can be defined with the periodic system object definition command

```
sys = ps(a,b,@c,0,2*pi);
```

To define a discrete-time periodic system of the form (2), the matrices with possibly time-varying dimensions are stored in cell arrays. For example, to store A_k for $k = 1, \ldots, N$, the kth element A_k is stored in $a\{k\}$ of the corresponding cell array a. An example of building a discrete-time periodic system with period $T = 10$ and discrete period $N = 2$ is

```
a = { [ 1 0 ], [ 1; 1] };
b = { [ 1 ], [ 1; 1] };
c = { [ 1 1 ], [ 1 ] };
d = { [ 1 ], [ 1 ]   };
sys = ps(a,b,c,d,10);
```

2.2 Building periodic system models

The toolbox allows to build periodic system models starting from various types of system data. For example, given the values of a periodic matrix $A(t)$ at equidistant time moments $A(t_i)$ for $t_i = i\frac{T}{n_t}$, $i = 0, \ldots n_t - 1$, a harmonic representation of $A(t)$ of the form (3) can be computed using

```
ah = ts2hr(a,nh);
```

where `a(:,:,i)` must contain $A(t_i)$. If `a(:,:,i)`, `b(:,:,i)`, `c(:,:,i)` contain $A(t_i)$, $B(t_i)$, $C(t_i)$ and $D(t) = 0$, then a periodic system object in harmonic representation can be generated with

```
sys = ps(ts2hr(a),ts2hr(b),ts2hr(c),0,T);
```

Another possibility to build a periodic system is simply by importing standard linear time-invariant systems. For example, if `SYS` = (A, B, C, D) is a continuous- or discrete-time linear time-invariant system, then a periodic system `PSYS` with period T can be generated with

```
PSYS = ps(SYS,T);
```

Discrete-time periodic systems can be generated by the discretization of a continuous-time periodic system. For example, if `PSYSC` = $(A(t), B(t), C(t), D(t))$ is a continuous-time linear time-varying periodic system (1) of period T, then the corresponding discrete-time system `PSYSD` of the form (2) for a sampling-time $T_s := T/N$ can be obtained with

```
PSYSD = psc2d(PSYSC,PSYSC.Period/N);
```

Periodic systems originate frequently from multi-rate discretization of standard linear time-invariant systems. Let `SYS` = (A, B, C, D) be a continuous- or discrete-time linear time-invariant system with p-outputs and m-inputs and let T_s be the basic sampling period. It is assumed that the jth input component is sampled with sampling period k_jT_s and the ith output component is sampled with sampling period l_iT_s, where k_j and l_i are integers. A periodic system describing this multirate sampled-data system can be obtained with

```
PSYS = psmrc2d(SYS,Ts,ks,ls);
```

where `ks` and `ls` are vectors containing k_j, $j = 1, \ldots, m$ and l_i, $i = 1, \ldots, p$, respectively.

2.3 Basic operations

The object-oriented framework allows to easily manipulate periodic models. Parallel and series couplings can be performed by "adding"/"subtracting" and "multiplying" two systems via the commands

```
ppar = ps1 + ps2;  % parallel coupling
pdif = ps1 - ps2;  % difference
pser = ps1 * ps2;  % series coupling
```

Furthermore, building an inverse system, a kind of dual system or performing time-shifting is extremely simple using commands like

```
psysinv  = psinv(psys);      % inverse
dpsys    = psdual(psys);     % dual system
spsys    = psshift(psys,k); % time-shifting
```

Part of these operations are available only for discrete-time systems. For these type of systems, recall that the system data are stored in cell arrays which can be easily associated with block diagonal matrices. Thus, many operations with block-diagonal matrices (e.g., sums, products, norms, etc.) can be implicitly performed via equivalent operations on cell arrays. The toolbox provides a rich collection of such tools for handling cell arrays with one-to-one correspondence to manipulating block diagonal matrices.

3. LIFTED REPRESENTATIONS

Lifted representations of discrete-time periodic systems play an important role in extending many theoretical results for standard systems to the periodic setting (Bittanti and Colaneri, 1996; Bittanti and Colaneri, 2000). They are also used to define concepts which corresponds to those for standard systems (e.g., transfer-function, poles, zeros, etc.). Although not well-suited for solving computational problems, lifted representations can still be used for algorithm development and testing in conjunction with appropriate tools able to manipulate large order standard or descriptor system representations (Varga, 2000*b*). This is why, a set of functions have been implemented to generate and handle several lifted representations.

3.1 Standard lifted representation

For the periodic system (2) the associated lifted time-invariant representation introduced in (Meyer and Burrus, 1975) uses the input-output behavior of the system over time intervals of length N, rather than 1. Let $M = \sum_{i=1}^{N} m_i$ and $P = \sum_{i=1}^{N} p_i$ be the sums of dimensions of input and output vectors over one period. For a given sampling time k, the corresponding M-dimensional input, P-dimensional output, and n_k-dimensional state vectors are

$$\begin{aligned} u_k^L(h) &= [u^T(k+hN) \cdots u^T(k+hN+N-1)]^T, \\ y_k^L(h) &= [y^T(k+hN) \cdots y^T(k+hN+N-1)]^T, \\ x_k^L(h) &= x(k+hN) \end{aligned}$$

Denote the transition matrix between time moments i and j by $\Phi_A(j,i) := A_{j-1} \cdots A_{i+1}A_i$, $\Phi_A(i,i) := I_{n_i}$. The *standard lifted system* has the form

$$\begin{aligned} x_k^L(h+1) &= F_k^L x_k^L(h) + G_k^L u_k^L(h) \\ y_k^L(h) &= H_k^L x_k^L(h) + L_k^L u_k^L(h) \end{aligned} \qquad (4)$$

where

$$F_k^L = \Phi_A(k+N,k)$$
$$G_k^L = [\ \Phi_A(k+N,k+1)B_k\ \cdots B_{k+N-1}\ ]$$
$$H_k^L = \begin{bmatrix} C_k \\ \vdots \\ C_{k+N-1}\Phi_A(k+N-1,k) \end{bmatrix}$$
$$L_k^L = \begin{bmatrix} D_k & 0 & \cdots & 0 \\ L_{k,2,1} & D_{k+1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{k,N,1} & L_{k,N,2} & \cdots & D_{k+N-1} \end{bmatrix}$$

with $L_{k,i,j} = C_{k+i-1}\Phi_A(k+i-1,k+j)B_{k+j-1}$, for $i = 2,...,N$, $j = 1,2,\ldots N-1$, and $i > j$.

The *transfer function matrix* (TFM) of the periodic system (2) at sampling time k is defined as the TFM of the lifted system (4)

$$W_k^L(z) = H_k^L(zI_{n_k} - F_k^L)^{-1}G_k^L + L_k^L \qquad (5)$$

The standard lifted representation `sys` of a periodic system `psys` can be computed with the command

```
sys = ps2ls(psys);
```

and the converse transformation is also possible via the command (N is the discrete period)

```
psys = ls2ps(sys,N);
```

The TFM `ltf` of a periodic system `psys` can be computed without explicitly forming the lifted representation (4) using the command

```
ltf = ps2tm(psys);
```

The underlying algorithm for this computation has been proposed in (Varga, 2003). Conversely, with the command

```
psys = tm2ps(ltf);
```

a minimal periodic realization `psys` of a lifted TFM `ltf` can be computed using the algorithm proposed in (Varga, 2004*d*).

3.2 Cyclic lifted representation

For notational convenience, a script notation will be used which associates to an N-periodic matrix X_k a block-diagonal matrix

$$\mathcal{X}_k := \mathrm{diag}\,(X_k, X_{k+1}, \ldots, X_{k+N-1})$$

The time-shifting of the periodic X_k is denoted by

$$\sigma\mathcal{X}_k := \mathrm{diag}\,(X_{k+1}, \ldots, X_{k+N-1}, X_k)$$

With

$$Z_k = \begin{bmatrix} 0 & \cdots & 0 & I_{n_{k+N-1}} \\ I_{n_k} & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & I_{n_{k+N-2}} & 0 \end{bmatrix}$$

the *cyclic lifted system* introduced in (Park and Verriest, 1989) has state-space dimension $\nu = \sum_{i=1}^{N} n_i$ and is defined by

$$x_k^C(h+1) = F_k^C x_k^C(h) + G_k^C u_k^C(h)$$
$$y_k^C(h) = H_k^C x_k^C(h) + L_k^C u_k^C(h)$$

where

$$(F_k^C, G_k^C, H_k^C, L_k^C) = (Z_k\mathcal{A}_k, Z_k\mathcal{B}_k, \mathcal{C}_k, \mathcal{D}_k)$$

The cyclic lifted representation `csys` of a periodic system `psys` can be computed with the command

```
csys = ps2ls(psys,'C');
```

3.3 Stacked lifted representation

The so-called *stacked lifted representation* introduced in (Grasselli and Longhi, 1991) uses

$$x_k^L(h) = [\ x^T(k+hN)\cdots x^T(k+hN+N-1)\ ]^T$$

as lifted state variable and is a time-invariant descriptor system representation of the form

$$E_k^S x_k^L(h+1) = F_k^S x_k^L(h) + G_k^S u_k^L(h)$$
$$y_k^L(h) = H_k^S x_k^L(h) + L_k^S u_k^L(h) \qquad (6)$$

where $G_k^S = Z_k\mathcal{B}_k$, $H_k^S = \mathcal{C}_k$, $L_k^S = \mathcal{D}_k$, and

$$F_k^S - zE_k^S = \begin{bmatrix} -zI_{n_k} & O & \cdots & O & A_{k+N-1} \\ A_k & -I_{n_{k+1}} & \cdots & O & O \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ O & O & \cdots & -I_{n_{k+N-2}} & O \\ O & O & \cdots & A_{k+N-2} & -I_{n_{k+N-1}} \end{bmatrix}$$

The TFM of the stacked lifted system is

$$W_k^S(z) = H_k^S(zE_k^S - F_k^S)^{-1}G_k^S + L_k^S$$

and it is easy to show that $W_k^S(z) = W_k^L(z)$, that is, the TFMs of the *stacked* and *standard lifted systems* are the same.

The stacked lifted representation `ssys` of a periodic system `psys` can be computed with the command

```
cssys = ps2ls(psys,'S');
```

4. PERIODIC SYSTEM ANALYSIS

The poles and zeros of a discrete-time periodic systems can be defined in terms of the associated lifted TFM $W_k^L(z)$. For a minimal realization, the poles can be computed as the eigenvalues of the monodromy matrix $\Phi_A(N+1,1)$ (also called characteristic multipliers) and for this purpose the periodic Hessenberg/Schur based reduction of the periodic matrix A_k is the appropriate method (Bojanczyk *et al.*, 1992; Hench and Laub, 1994). The basic reduction has been

implemented in Fortran 95 routines called from MATLAB via the *mex*-function `pschur`. For zeros computation, the "fast" structure exploiting algorithm of (Varga and Van Dooren, 2003) is able to compute both finite zeros and infinite zeros structure. These algorithms are implemented in the functions `pspole` and `pszero`, respectively. The characteristic exponents of a periodic matrix $A(t)$ or the characteristic multipliers of a periodic sequence A_k can be computed by using the functions `psceig` and `pseig`, respectively.

For non-minimal systems, the input- and output-decoupling zeros can be computed with the functions `psidzero` and `psodzero`, respectively. To compute minimal realizations, the function `psminreal` can be employed. This function is based on computing the periodic reachability and observability Kalman forms via the algorithm proposed in (Varga, 2004*b*). For efficiency purposes, the structure preserving reduction used in this algorithm has been implemented in Fortran 95 and called from MATLAB via the *mex*-function `psystr`. For order reduction of stable periodic systems the functions `psbtabal` and `psbta` implementing the *square-root* and *balancing-free square-root* balanced truncation approximation technique of (Varga, 2000*a*) can be employed. The function `psminrealbal` can be employed to compute minimal realizations for periodic systems using the balanced truncation approach.

5. PERIODIC MATRIX EQUATION SOLVERS

Solvers for both the reverse-time periodic Lyapunov equation

$$\mathcal{X} = \mathcal{A}^T \sigma\mathcal{X}\mathcal{A} + \mathcal{C}$$

as well as for the forward-time periodic Lyapunov equation

$$\sigma\mathcal{X} = \mathcal{A}\mathcal{X}\mathcal{A}^T + \mathcal{C}$$

are provided via the functions `prlyap` and `pflyap`, implementing the algorithms of (Varga, 1997) extended to time-varying dimensions. These equations appear when solving periodic state-feedback stabilization problems, computing gradients for periodic output feedback optimization, or computing special coprime factorizations. These functions rely on Fortran 95 implementations of the solution algorithms called via the *mex*-funtion `pslinmeq`.

For the non-negative periodic Lyapunov equations in reverse-time

$$\mathcal{R}^T\mathcal{R} = \mathcal{A}^T\sigma(\mathcal{R}^T\mathcal{R})\mathcal{A} + \mathcal{C}^T\mathcal{C}$$

and in forward-time

$$\sigma(\mathcal{S}\mathcal{S}^T) = \mathcal{A}(\mathcal{S}\mathcal{S}^T)\mathcal{A}^T + \mathcal{B}\mathcal{B}^T$$

iterative solvers, proposed in (Varga, 1997), have been implemented to compute the Cholesky-factors of the reachability and observability grammians. The corresponding function `psgram` underlies the computation of periodic gramians used for internal balancing and balancing based model reduction in the functions `psminrealbal`, `psbtabal` and `psbta`.

Solvers for the periodic Riccati equations in reverse-time

$$\mathcal{X} = \mathcal{A}^T[\sigma\mathcal{X} - \sigma\mathcal{X}\mathcal{B}(\mathcal{R} + \mathcal{B}^T\sigma\mathcal{X}\mathcal{B})^{-1}\mathcal{B}^T\sigma\mathcal{X}]\mathcal{A} + \mathcal{Q}$$

or in forward-time

$$\sigma\mathcal{X} = \mathcal{A}[\mathcal{X} - \mathcal{X}\mathcal{C}^T(\mathcal{R} + \mathcal{C}\mathcal{X}\mathcal{C}^T)^{-1}\mathcal{C}\mathcal{X}]\mathcal{A}^T + \mathcal{B}\mathcal{B}^T$$

are implemented as the functions `prdare` and `pfdare`, respectively. The basic solvers use a structure exploiting "fast" algorithm proposed in (Varga, 2005) to compute orthogonal bases of the stable deflating subspace of a large lifted pencil without forming explicitly this pencil. These functions are useful to solve periodic LQ-design and filtering problems.

6. LINEAR-QUADRATIC STABILIZATION

Minimizing a linear-quadratic criterion of the form

$$J = \sum_{k=0}^{\infty} [\, x_k^T Q_k x_k + u_k^T R_k u_k \,] \qquad (7)$$

where $Q_k \geq 0$, $R_k > 0$ are N-periodic symmetric matrices, represents an attractive method to determine stabilizing periodic feedback controllers. The optimal periodic state-feedback matrix F_k in the control law

$$u_k^* = F_k x_k$$

which minimizes the performance index (7) can be determined using the functions `plqr` and `plqry`, where the latter uses output-weighting $y_k^T Q_k y_k$ in (7) instead of state-weighting $x_k^T Q_k x_k$.

The available parametric freedom when designing periodic controllers can be conveniently exploited by designing output feedback controllers. The optimal periodic output-feedback gain F_k in the control law

$$u_k^* = F_k y_k$$

which minimizes the performance index (7) can be computed using the function `plqofc`. This function is based on a gradient-based function minimization technique for large scale problems (limited memory BFGS with simple bounds), where the function and gradient evaluations are implemented as a Fortran 95 *mex*-function based on the formulas derived in (Varga and Pieters, 1998). A variant of this function, `plqocfc`, allows to compute optimal constant output feedback gain matrices for a periodic system.

7. PLANNED NEW DEVELOPMENTS

The primary goal for developing the PERIODIC SYSTEMS Toolbox for MATLAB was to provide tools for manipulating linear periodic systems which arise in several important application areas. The toolbox provides a basic functionality for discrete-time periodic systems which can be seen as a direct extension of functions provided in the MATLAB Control Toolbox. By relying on structure exploiting and structure preserving numerically stable algorithms (many of them developed in the last few years), this toolbox is the first CACSD tool able to handle realistic problems where the system order and/or the period are large. A first application of this toolbox is the solution of a challenging attitude control problem (period $N = 500$) for a magnetically actuated satellite using periodic output feedback control (Lovera and Varga, 2005).

There are many planned developments of this toolbox, which are conditioned by the progress in developing new algorithms. One direction is to extend the algorithmic basis for discrete-time periodic systems by implementing algorithms for reordering periodic Schur forms, periodic QZ, computing periodic Kronecker-like forms, solving periodic Sylvester equations, as well as developing reliable algorithms for solving stabilization, factorizations, norms computation problems. A major effort will be to implement recently developed numerical algorithms for continuous-time periodic systems (e.g., solving differential periodic Lyapunov, Sylvester and Riccati equations (Varga, 2005*b*)). It is envisaged to extend the support for object-oriented manipulation of models based on harmonic representations (e.g., couplings) as well as improving the algorithm for harmonic approximation of time series. Besides adding time simulations tools (step-response, impulse-response, etc.), developing efficient algorithms and tools for frequency-domain techniques will be a new research focus. The increasing importance of multi-rate control systems requires an extended support for various sample and hold devices and dedicated synthesis methodologies which completely avoid the building of lifted representations.

REFERENCES

Bittanti, S. and P. Colaneri (1996). Analysis of discrete-time linear periodic systems. In *Control and Dynamics Systems.* (C. T. Leondes, Ed.). Vol. 78, pp. 313–339. Academic Press.

Bittanti, S. and P. Colaneri (2000). Invariant representations of discrete-time periodic systems. *Automatica* **36**, 1777–1793.

Bojanczyk, A. W., G. Golub and P. Van Dooren (1992). The periodic Schur decomposition. Algorithms and applications. *Proceedings SPIE Conference* (F. T. Luk, Ed.). Vol. 1770. pp. 31–42.

Grasselli, O. M. and S. Longhi (1991). Pole-placement for nonreachable periodic discrete-time systems. *Math. Control Signals Syst.* **4**, 439–455.

Hench, J. J. and A. J. Laub (1994). Numerical solution of the discrete-time periodic Riccati equation. *IEEE Trans. Automat. Control* **39**, 1197–1210.

Lovera, M. and A. Varga (2005). Optimal discrete-time magnetic attitude control of satellites. Prepr. IFAC 2005 World Congress, Prague, Czech Republic.

Meyer, R. A. and C. S. Burrus (1975). A unified analysis of multirate and periodically time-varying digital filters. *IEEE Trans. Circuits Syst.* **22**, 162–168.

Park, B. and E. I. Verriest (1989). Canonical forms for discrete-time periodically time varying systems and a control application. *Proc. of CDC'89, Tampa, Florida*, pp. 1220–1225.

Varga, A. (1997). Periodic Lyapunov equations: some applications and new algorithms. *Int. J. Control* **67**, 69–87.

Varga, A. (2000*a*). Balanced truncation model reduction of periodic systems. *Proc. of CDC'2000, Sydney, Australia*. pp. 2379–2384.

Varga, A. (2000*b*). A DESCRIPTOR SYSTEMS Toolbox for MATLAB. *Proc. of CACSD'2000 Symposium, Anchorage, Alaska*.

Varga, A. (2003). Computation of transfer functions matrices of periodic systems. *Int. J. Control* **76**, 1712–1723.

Varga, A. (2004*a*). Computation of generalized inverses of periodic systems. *Proc. of CDC'04, Paradise Island, Bahamas*.

Varga, A. (2004*b*). Computation of Kalman decompositions of periodic systems. *European Journal of Control* **10**, 1–8.

Varga, A. (2004*c*). Computation of Kronecker-like forms of periodic matrix pairs. *Proc. of MTNS'04, Leuven, Belgium*.

Varga, A. (2004*d*). Computation of minimal periodic realizations of transfer-function matrices. *IEEE Trans. Automat. Control* **46**, 146–149.

Varga, A. (2004*e*). Design of fault detection filters for periodic systems. *Proc. of CDC'04, Paradise Island, Bahamas*.

Varga, A. (2005). On solving discrete-time periodic Riccati equations. Prepr. of IFAC 2005 World Congress, Prague, Czech Republic.

Varga, A. (2005*b*). On solving periodic differential matrix equations. *(submitted to CDC 2005, Seville, Spain)*.

Varga, A. and P. Van Dooren (2001). Computational methods for periodic systems - an overview. *Proc. of IFAC Workshop on Periodic Control Systems, Como, Italy*, pp. 171–176.

Varga, A. and P. Van Dooren (2003). Computing the zeros of periodic descriptor systems. *Systems & Control Lett.* **50**, 371381.

Varga, A. and S. Pieters (1998). Gradient-based approach to solve optimal periodic output feedback control problems. *Automatica* **34**, 477–481.