Computing the Kalman form

Clément Pernet, Aude Rondepierre, Gilles Villard

Computing the Kalman form

Clément Pernet
LMC, Université Joseph Fourier
51, rue des Mathématiques BP 53 IMAG-LMC 38041 Grenoble, FRANCE
clement.pernet@imag.fr

Aude Rondepierre
LMC, Université Joseph Fourier
51, rue des Mathématiques BP 53 IMAG-LMC 38041 Grenoble, FRANCE
aude.rondepierre@imag.fr

Gilles Villard
CNRS, LIP, Ecole Normale Supérieure de Lyon
46, Allée d'Italie, 69364 Lyon Cedex 07 FRANCE
gilles.villard@ens-lyon.fr

October 18, 2005

Abstract

We present two algorithms for the computation of the Kalman form of a linear control system. The first one is based on the technique developed by Keller-Gehrig for the computation of the characteristic polynomial. The cost is a logarithmic number of matrix multiplications. To our knowledge, this improves the best previously known algebraic complexity by an order of magnitude. Then we also present a cubic algorithm proven to more efficient in practice.

1 Introduction

This report is a continuation of a collaboration with Aude Rondepierre on the algorithmic similarities between the computation of the Kalman form and of the characteristic polynomial. This collaboration led to [DR05, Theorem 2]. We report here an improvement of this last result based on a remark by the second author.

For a definition of the Kalman form of a linear control system, see [Kal61, Theorem 1]. In this report we show how to adapt the branching algorithm of Keller-Gehrig [KG85, §5] (computing the characteristic polynomial) to compute the Kalman form. This implies an algebraic time complexity of $\mathcal{O}(n^\omega \log n)$. Now, the discussion of [DPW05, §2] shows that a cubic algorithm, `LUK`, is more efficient in practice for the computation of the characteristic polynomial. Therefore, we adapt it to the computation of the Kalman form.

The outline of this report is the following : in section 2 we recall the principle of the branching algorithm of Keller-Gehrig. Then we decompose the computation of the Kalman form into two steps : the computation of the so-called "compressed Krylov matrix" and the computation of the Kalman form itself. For the first step, the algorithm of Keller-Gehrig is easily adapted in 3.1.1. Alternatively the same computation can be performed by an adaptation of the algorithm `LUK` as shown in 3.1.2. Now, the recovery of the Kalman form from this compressed Krylov matrix also uses the technique of `LUK` algorithm in section 3.2.1. Lastly, algorithm 3.2.2 merges the ideas of sections 3.1.2 and 3.2.1 but removes some unnecessary operations.

2 Keller-Gehrig's branching algorithm

2.1 Principle

Consider the $n \times n^2$ Krylov matrix K generated by the n column vectors of the identity matrix iterated with the matrix A (of order n) :

$$K = \left[\; e_1 \; \middle| \; \ldots \; \middle| \; A^{n-1}e_1 \; \middle| \; \ldots \; \middle| \; e_n \; \middle| \; \ldots \; \middle| \; A^{n-1}e_n \; \right]$$

The rank of K is n. Let us form the $n \times n$ non-singular matrix $\overline{K}$ by picking the first n linearly independent rows of K.

If a column vector $A^k e_j$ is linearly dependent with the previous iterates, then any vector $A^l e_j, l > k$ will also be linearly dependent. Consequently the matrix $\overline{K}$ has the form :

$$\overline{K} = \left[\; e_1 \; \middle| \; \ldots \; \middle| \; A^{d_1}e_1 \; \middle| \; \ldots \; \middle| \; e_m \; \middle| \; \ldots \; \middle| \; A^{d_m}e_m \; \right] \tag{1}$$

with $0 \leq d_i \leq n-1$.

Property 1. *The matrix $\overline{K}^{-1} A \overline{K}$ has the Hessenberg polycyclic form : it is block upper triangular, with companions blocks on its diagonal, and the upper blocks are zero except on their last column.*

$$\overline{K}^{-1} A \overline{K} = \begin{bmatrix} \boxed{\begin{matrix} 0 & & & * \\ 1 & 0 & & * \\ & \ddots & \ddots & * \\ & & 1 & * \end{matrix}} & & & * \\ & & & * \\ & & & * \\ & & & * \\ & \ddots & & \\ & & \boxed{\begin{matrix} 0 & & & * \\ 1 & 0 & & * \\ & \ddots & \ddots & * \\ & & 1 & * \end{matrix}} & \end{bmatrix} \tag{2}$$

Corollary 1. *The characteristic polynomial of A is the product of the polynomials associated with the companion blocks of the diagonal of H.*

2.2 The algorithm

Ingredients :

- The position of the linearly independent columns is given by a gaussian elimination. A block elimination is mandatory to reduce the algebraic complexity to matrix multiplication. Keller-Gehrig introduced in [KG85, §4] an algorithm called "step form elimination". The more recent litterature replaced it by the row echelon elimination (for example in [CBS97]). We showed in [DPW05] that the `LQUP` (defined by in [IMH82]) elimination of $\overline{K}^T$ (algorithm 2.2.1) could also be used.

Algorithm 2.2.1 `ColReducedForm`

Require: A a $m \times n$ matrix of rank r ($m, n \geq r$) over a field
Ensure: A' a $m \times r$ matrix formed by r linearly independent columns of A
1: $(L, Q, U, P, r) =$ LQUP(A^T) ($r = rank(A)$)
2: return $([I_r 0](Q^T A^T))^T$

- The computation of the matrix K is prohibitive (n^3 coefficients and $\mathcal{O}(n^4)$ arithmetic operations with standard matrix product). Hence, the elimination process must be combined within the building of the matrix.

- The computation of the iterates can rely on matrix multiplication, by computing the $\lceil \log_2(n) \rceil$ following powers of A :

$$A, A^2, \ldots, A^{2^i}, A^{2^{\lceil \log_2(n) \rceil - 1}}$$

Thus the following scheme

$$\left\{ \begin{array}{lcl} V_0 & = & [e_j] \\ V_{i+1} & = & [V_i | A^{2^i} V_i] \end{array} \right. \quad (3)$$

computes every iterates of e_j in $\mathcal{O}(n^\omega \log n)$ operations.

- One elimination is performed after each application of A^{2^i}, to discard the linearly dependent iterates for the next iteration step. Moreover if a vector e_j has only $k < 2^i$ linearly independent iterates, one can stop the computation of its iterates. Therefore, the scheme (3) will only be applied on the block iterates of size 2^i.

The algorithm is the following :

Algorithm 2.2.2 Characteristic Polynomial [Keller-Gehrig]

Require: A a $n \times n$ matrix over a field
Ensure: P^A_{char} the characteristic polynomial of A
1: $i = 0$; $V_0 = I_n = [V_{0,1} | V_{0,2} | \ldots | V_{0,n}]$
2: $B = A$
3: **while** ($\exists k, V_{i,k}$ has 2^i columns) **do**
4: **for all** j **do**
5: **if** ($V_{i,j}$ has strictly less than 2^i columns) **then**
6: $W_j = V_{i,j}$
7: **else**
8: $W_j = [V_{i,j} | B V_{i,j}]$
9: **end if**
10: **end for**
11: $W = [W_1 | \ldots | W_n]$
12: V_{i+1} = ColReducedForm(W) $\{V_{i+1} = [V_{i+1,1} | \ldots | V_{i+1,n}]$ where $V_{i+1,j}$ are the remaining vectors of W_j in $V_{i+1}\}$
13: $B = B \times B$; $i = i + 1$
14: **end while**
15: $F = V^{-1} A V$, F is as in (2)
16: $f = \Pi_i f_i$ where f_i is the polynomial of the ith companion block of F
17: return f

3 Application to the computation of the Kalman form

In the following, we will show that the algebraic time complexity for the computation of the Kalman form given in [DR05, Theorem 2] can be improved by an order of magnitude by adapting the branching algorithm of Keller-Gehrig. Let us decompose the computation of the Kalman form into two steps :

- the computation of the compressed Krylov matrix $\overline{K}$ from equation 1 (part 3.1).
- The computation of the Kalman form itself (part 3.2.1)

The first step can be done using either Keller-Gehrig technique (section 3.1.1) or using `LUK` (section 3.1.2). In both situations, some computations can be saved by merging the two steps, as we do in section 3.2.2, since they share common operations.

3.1 Computation of the compressed Krylov matrix

3.1.1 A la Keller-Gehrig

Algorithm 3.1.1 `KGCKM` Compressed Krylov Matrix (Keller-Gehrig)

Require: A a $n \times n$ matrix over a field, B, a $n \times m$ matrix
Ensure: $(V, rank(V))$ as in (1)

1: $i = 0$
2: $V_0 = B = (V_{0,1}, V_{0,2}, \ldots, V_{0,m})$
3: $C = A$
4: **while** ($\exists k, V_k$ has 2^i columns) **do**
5: **for all** j **do**
6: **if** ($V_{i,j}$ has strictly less than 2^i columns) **then**
7: $W_j = V_{i,j}$
8: **else**
9: $W_j = [V_{i,j}|CV_{i,j}]$
10: **end if**
11: **end for**
12: $W = [W_1|\ldots|W_n]$
13: V_{i+1} = ColReducedForm(W) remember $r = rank(W)$ $\{V_{i+1} = [V_{i+1,1}|\ldots|V_{i+1,n}]$ where $V_{i+1,j}$ are the remaining vectors of W_j in $V_{i+1}\}$
14: $C = C \times C$
15: $i = i + 1$
16: **end while**
17: return (V_i, r)

The idea in algorithm 3.1.1 is just to replace the identity matrix I_n by the matrix B in algorithm 2.2.2. Then, the last computed V_i is formed by some column vectors of B and their iterates with the matrix A. This matrix is a basis of Span$(B, AB, \ldots, A^{n-1}B)$.

3.1.2 A la LU-Krylov

Algorithm 3.1.2 presents another method to compute $\overline{K}$, adapted from [DPW05, algorithm 2.2]. This algorithm is not as good in theory ($\mathcal{O}(n^3)$ versus $\mathcal{O}(n^\omega \log n)$), but proven to be more efficient in practice [DPW05, §2.4]. Its correctness can be proven in the same way as for algorithm 3.2.1.

3.2 Computation of the kalman form

3.2.1 From the compressed Krylov matrix

Using the technique of algorithm `LUK` ([DPW05, theorem 2.1]), the matrix T can be recovered by completing the compressed Krylov matrix $\overline{K}$ into an invertible matrix. This can be easily done by a triangularization of $\overline{K^T}$ as follows : one computes the LUP factorization of $\overline{K}^T$, and replace $[U_1 U_2]$ by $\begin{bmatrix} U_1 & U_2 \\ 0 & Id \end{bmatrix}$ and $[L]$ by $\begin{bmatrix} L & 0 \\ 0 & Id \end{bmatrix}$. This simply corresponds to set

$$T = \left[\begin{array}{c|c} \overline{K} & P^T \begin{bmatrix} 0 \\ I_{n-r} \end{bmatrix} \end{array} \right].$$

Theorem 1. *Let A and B be two matrices of dimension respectively $n \times n$ and $n \times m$. Let r be the rank of Span$(B, AB, \ldots, A^{n-1}B)$. Algorithm 3.2.1 computes the Kalman form :*

$$T^{-1}AT = \begin{bmatrix} H & C_1^T \\ 0 & C_2^T \end{bmatrix}, \begin{bmatrix} B_1 \\ 0 \end{bmatrix} = T^{-1}B$$

Algorithm 3.1.2 `LUKCKM` : Compressed Krylov Matrix (LU-Krylov)

Require: A a $n \times n$ matrix over a field, B, a $n \times m$ matrix
Ensure: $(V, rank(V))$ as in (1)

$v = B_1$
$\begin{cases} K = \begin{bmatrix} v & Av & A^2v & \dots \end{bmatrix} \\ (L, [U_1|U_2], P) = \texttt{LUP}(K^T), r_1 = \text{rank}(K) \end{cases}$ {The matrix K is computed on the fly : at most $2r_1$ columns are computed. U_1 is $r_1 \times r_1$}
if $(r_1 = n)$ **then**
  return (K, r_1)
else
  $A' = PAP^T = \begin{bmatrix} A'_{11} & A'_{12} \\ A'_{21} & A'_{22} \end{bmatrix}$ where A'_{11} is $r_1 \times r_1$.
  $A_R = A'_{22} - U_2^T U_1^{-T} A'_{12}$
  $B' = \begin{bmatrix} L^{-T} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} U_1^{-T} & 0 \\ -U_2^T U_1^{-T} & I \end{bmatrix} PB$
  Compute the permutation Q s.t. $B'Q = \begin{bmatrix} X & Y \\ 0 & Z \end{bmatrix}$
  $(V_2, r_2) = \texttt{LUCKM}(A_R, Z)$
  $V = \left[\begin{array}{c|c} K & P^T \begin{bmatrix} 0 \\ V_2 \end{bmatrix} \end{array}\right]$
  return $(V, r_1 + r_2)$
end if

Algorithm 3.2.1 Kalman form

Require: A a $n \times n$ matrix over a field, B, a $n \times m$ matrix
Ensure: r, T, H, C_1, C_2, B_1 as in theorem 1

$(V, r) = \texttt{CompressedKrylovMatrix}(A, B)$
if (r=n) **then**
  return $(n, Id, A, \emptyset, \emptyset, B)$
else
  $(L, [U_1 U_2], P) = \text{LUP}(V^T)$
  $T = \left[\begin{array}{c|c} V & P^T \begin{bmatrix} 0 \\ I_{n-r} \end{bmatrix} \end{array}\right]$
  $B_1 = L^{-T} U_1^{-T} PB$
  $A' = PA^T P^T = \begin{bmatrix} A'_{11} & A'_{12} \\ A'_{21} & A'_{22} \end{bmatrix}$
  $C_1 = L^{-T} U_1^{-T} A'_{12}$
  $C_2 = A'_{22} - U_2^T U_1^{-T} A'_{12}$
  for all j **do**
    Let t_j be the column indexes in V of the last iterate of the jth block.
    Let l_j be first linearly dependent row of the jth block in L (to be stored during the execution of `CompressedKrylovMatrix`)
    $m_j = l_j L^{-1}_{1 \dots t_j, 1 \dots t_j}$
  end for
  Build the polycyclic matrix H by placing each column vectors m_j at column index t_j and adding 1 on the subdiagonal on all other columns.
  return (r, T, H, C_1, C_2, B_1)
end if

where H and B_1 are respectively $r \times r$ et $r \times m$.

It requires $\mathcal{O}(n^\omega log n)$ field operations using algorithm 3.1.1 or $\mathcal{O}(n^3)$ using algorithm 3.1.2.

Proof. Algorithm `CompressedKrylovMatrix` builds the matrix V satisfying

$$AV = VH$$

where H has the Hessenberg polycylic form (2).

Now

$$\begin{aligned} AT &= \left[\begin{array}{c|c} AV & AP^T \begin{bmatrix} 0 \\ I_{n-r} \end{bmatrix} \end{array}\right] \\ &= T \begin{bmatrix} H & C_1 \\ 0 & C_2 \end{bmatrix} \end{aligned}$$

with $T \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = AP^T \begin{bmatrix} 0 \\ I_{n-r} \end{bmatrix}$. Let us write

$$A' = PAP^T = \begin{bmatrix} A'_{11} & A'_{12} \\ A'_{21} & A'_{22} \end{bmatrix}.$$

We have

$$T \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = P^T A' \begin{bmatrix} 0 \\ I_{n-r} \end{bmatrix} = P^T \begin{bmatrix} A'_{12} \\ A'_{22} \end{bmatrix}$$

Now

$$T = P^T \begin{bmatrix} U_1^T & 0 \\ U_2^T & I_{n-r} \end{bmatrix} \begin{bmatrix} L^T & 0 \\ 0 & I_{n-r} \end{bmatrix},$$

therefore

$$\begin{aligned} \begin{bmatrix} U_1^T & 0 \\ U_2 & I_{n-r} \end{bmatrix} \begin{bmatrix} L^T & 0 \\ 0 & I_{n-r} \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} &= \begin{bmatrix} A'_{12} \\ A'_{22} \end{bmatrix} \\ \begin{bmatrix} U_1^T & 0 \\ U_2^T & I_{n-r} \end{bmatrix} \begin{bmatrix} L^T C_1 \\ C_2 \end{bmatrix} &= \begin{bmatrix} A'_{12} \\ A'_{22} \end{bmatrix} \end{aligned}$$

And the system

$$\left\{ \begin{array}{lcl} U_1^T L^T C_1 & = & A'_{12} \\ U_2^T L^T C_1 + C_2 & = & A'_{22} \end{array} \right.$$

has the following solution

$$\left\{ \begin{array}{lcl} C_1 & = & L^{-T} U_1^{-T} A'_{12} \\ C_2 & = & A'_{22} - U_2^T U_1^{-T} A'_{12} \end{array} \right.$$

Lastly

$$T^{-1} A T = \left[\begin{array}{cc} H & C_1 \\ 0 & C_2 \end{array}\right],$$

The column vectors of B are linear combinations of the column vectors of V, therefore we can write

$$B = T \left[\begin{array}{c} B_1 \\ 0 \end{array}\right]$$

and then

$$B_1 = L^{-T} U_1^{-T} PB.$$

If `CompressedKrylovMatrix` is done by algorithm 3.1.1, it requires $\mathcal{O}(n^\omega \log n)$ field operations : at most $\log n$ executions of the `while` loop whose domining operation is $B = B \times B$ in n^ω. Any remaining operation can be done in $\mathcal{O}(n^\omega)$.

Yet if `CompressedKrylovMatrix` is done by algorithm 3.1.2, the Krylov matrix is computed by at most n matrix-vector products ($\mathcal{O}(n^2)$ each) and any remaing operation can be done in $\mathcal{O}(n^3)$ using classic matrix arithmetic.

□

3.2.2 Improvements

Using Keller-Gehrig variant : One can note that the LUP factorization of V is already available at the end of algorithm 3.1.1. Thus, step 5 in algorithm 3.2.1 can be skipped.

Using `LUK` variant : Here again, one can merge the common operations between algorithms 3.1.2 and 3.2.1 into algorithm 3.2.2. More precisely, every operation is already done in algorithm 3.1.2 except the update of the upper right hand side block of H (denoted by J_1 in algorithm 3.2.2). However, we intentionnaly forgot to explain how to compute it in the previous combination of algorithms 3.1.2 and 3.2.1 and precise it here. Note that this computation of J_1 should be improved, taking into account its structure : a few non zero columns, since it is a submatrix of the Hessenberg polycyclic matrix H.

Algorithm 3.2.2 `Kalman-LUK` : Kalman-LU-Krylov

Require: A a $n \times n$ matrix over a field, B, a $n \times m$ matrix
Ensure: r_1, T, H, C_1, C_2, B_1 as in theorem 1

$v = B_1$
$\begin{cases} K = \begin{bmatrix} v & Av & A^2v & \dots \end{bmatrix} \\ (L, [U_1|U_2], P) = \texttt{LUP}(K^T), r_1 = \text{rank}(K) \end{cases}$ {The matrix K is computed on the fly : at most $2r_1$ columns are computed}
$m = (m_1, \dots, m_{r1}) = L_{r_1+1} L^{-1}_{1\dots r_1}$
$f = X^{r_1} - \sum_{i=1}^{r_1} m_i X^{i-1}$
if $(r_1 = n)$ **then**
  return $(n, Id, A, \emptyset, \emptyset, B)$
else
  $A' = PAP^T = \begin{bmatrix} A'_{11} & A'_{12} \\ A'_{21} & A'_{22} \end{bmatrix}$ where A'_{11} is $r_1 \times r_1$.
  $A_R = A'_{22} - U_2^T U_1^{-T} A'_{12}$
  $B' = \begin{bmatrix} L^{-T} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} U_1^{-T} & 0 \\ -U_2^T U_1^{-T} & I \end{bmatrix} PB$
  Compute the permutation Q s.t. $B'Q = \begin{bmatrix} X & Y \\ 0 & Z \end{bmatrix}$ $\{Z$ is $\mu \times \mu\}$
  if $(\mu = 0)$ **then**
    $C_1 = L^{-T} U_1^{-T} A'_{12}$
    $C_2 = A_R$
    $T = \left[\begin{array}{c|c} K & P^T \begin{bmatrix} 0 \\ I_{n-r_1} \end{bmatrix} \end{array} \right]$
    return $(r_1, T, C_f, C_1, C_2, X)$
  else
    $(r_2, T^{(2)}, H^{(2)}, C_1^{(2)}, C_2^{(2)}, B_1^{(2)}) = \texttt{Kalman-LUK}(A_R, Z)$
    $T = \left[\begin{array}{c|c} K & P^T \begin{bmatrix} 0 \\ T^{(2)} \end{bmatrix} \end{array} \right]$
    $J = L^{-T} U_1^{-T} A'_{12} T^{(2)} = \begin{bmatrix} J_1 & J_2 \end{bmatrix}$ $\{J_1$ is $r_1 \times r_2$ and J_2, $r_1 \times (n - r_1 - r_2)\}$
    $H = \begin{bmatrix} C_f & J_1 \\ 0 & H^{(2)} \end{bmatrix}$
    $C_1 = \begin{bmatrix} J_2 \\ C_1^{(2)} \end{bmatrix}$
    return $(r_1 + r_2, T, H, C_1, C_2^{(2)}, B_1)$
  end if
end if

Algorithm 3.2.2 is still $\mathcal{O}(n^3)$ since the additional operations are the following :

- the computation of L^{-T} in $\mathcal{O}(r_1^3)$
- the matrix product $(L^{-T}U_1^{-T}A'_{12}) \times T^{(2)}$ in $\mathcal{O}(r_1(n-r_1)^2)$

Summing for every recursive level, one gets $\mathcal{O}(n^3)$ since the sum of the r_1 of each recursive level is n.

References

[CBS97] Michael Clausen, Peter Burgisser, Mohammad A. Shokrollahi. *Algebraic Complexity Theory*. 1997.

[DPW05] Jean-Guillaume Dumas, Clément Pernet, Zhendong Wan. Efficient computation of the characteristic polynomial. Patrz Kauers [Kau05].

[DR05] Jean-Guillaume Dumas, Aude Rondepierre. Algorithms for symbolic/numeric control of affine dynamical system. Patrz Kauers [Kau05].

[IMH82] Oscar H. Ibarra, Shlomo Moran, Roger Hui. A generalization of the fast LUP matrix decomposition algorithm and applications. *Journal of Algorithms*, 3(1):45–56, Marzec 1982.

[Kal61] R.E. Kalman. Canonical structure of linear dynamical systems. *Proceedings of the National Academy of Sciences*, strony 596–600, 1961.

[Kau05] Manuel Kauers, redaktor. *ISSAC'2005. Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation, Beijing, China*. ACM Press, New York, Lipiec 2005.

[KG85] Walter Keller-Gehrig. Fast algorithms for the characteristic polynomial. *Theoretical computer science*, 36:309–317, 1985.