MILITARY TECHNICAL COLLEGE

CAIRO - EGYPT

Arnoldi Method for Transfer Function Matrices Computation of Linear Multivariable Control Systems.

H. NASR* M. RAMADAN**

ABSTRACT

We propose an Arnoldi-based numerical method to compute the transfer function matrices of linear | maultivariable system. The method is composed only of basic linear algebraic operations such as matrix-vector multiplication and solution of upper Hessenberg linear systems of smaller dimensions. The efficiency, compared with other methods, is shown by means of operations Counts. An illustrative numerical example is given to demonstrate the accuracy compared with the direct method.

* Prof. of Mathematics, Benha Higher Institute of Technology, Benha, Egypt.

** Dept. of Mathematics, Faculty of Science, Menoufia University.

I- INTRODUCTION

Consider the linear time-invariant system

$$\dot{x}(t) = A\,x(t) + B\,u(t)$$
$$y(t) = C\,x(t) + D\,u(t) \qquad (1.1)$$

where $x(t)$ is an n-vector of states, $u(t)$ is a p-vector of inputs and $y(t)$ is an m-vector of outputs. A,B,C and D are matrices of proper dimensions. The design and analysis of time-invariant linear control systems give rise to variety of interesting linear algebra problems. One of these problems is the computation of the transfer function matrix which is defined by,

$$G(s) = C\,(sI_n - A)^{-1}\,B + D \qquad (1.2)$$

The computation of the matrix $G(s)$ has been studied in control literature. Among the recent known methods are : Pole-Zero approach [3,7] and the approach in [5] which is based on some determinant identities and characteristic polynomial computation of an upper Hessenberg matrices. The above methods compute $G(s)$ one element at a time (single input - single output subsystems (A,b_j,c_i^T,d_{ij})), where each element is written in terms of denominator and numerator polynomials. They first require a reduction to Hessenberg form, for each input-output pair, to obtain the corresponding minimal order subsystem. Besides the computational effort of the reduction, it is known [4] that for large and sparse matrices the reduction to Hessenberg form (for example using orthogonal transformation) is not numerically effective.

In [3,7] the coefficients of the denominator polynomial, for each (i,j)- element, is obtained by computing the eigenvalues of the state matrix of the corresponding subsystem. The coefficients of the corresponding numerator polynomial is obtained by solving a generalized eigenvalue problem. Solving algebraic or generalized eigenvalue problem not only expensive, but also for systems with ill-conditioned eigenvalue or generalized eigenvalue problem the computed matrix G(S) will be inaccurate.

The, recently, efficient method proposed in [5] writes the denominator and numerator polynomials in terms of two determinants of upper Hessenberg matrices. Using a variant of Hymand's method [8] the two determinants computation is accomplished in approximatly $\frac{1}{3}n_c^3 + n_c$ (where n_c is the dimension of the corresponding controllable and observable subsystem).

In this paper, we propose an efficient method for computing the transfer function matrix based on Arnoldi - reduction in the initial process. A solution of upper Hessenberg linear systems of the same dimensions is followed to obtain the matrix G(s) one column at a time.

Since Arnoldi's algorithm uses matrix-vector multiplication only, then it is suitable for large and sparse matrices [4]. For each column vector computation, the method consists of choosing an initial Arnoldi-vector appropriately and running m-steps in the Arnoldi-algorithm. Finally, a solution of an m x m Hessenberg system followed by a matrix-vector multiplication to obtain the column vector. Besides the method is effective for large and sparse matrices, a certain amount of parallelism is involved which makes it suitable for parallel implementation.

II- THE PROPOSED ALGORITHM

We are concerned with an efficient computation of the transfer function matrix defined in (1.2). First, we drop the matrix D and in the final step we add it to the transfer function matrix. Therefore, our problem is reduced to the computation of the matrix,

$$G(s) = C\,(sI_n - A)^{-1}\,B.$$

The algorithm evaluates G(s) one column at a time, i.e. we evaluate,

$$g_i(s) := C\,(sI_n - A)^{-1}\,b_i, \quad i = 1,2,\ldots,p.$$

We shall describe first the Arnoldi-algorithm and state a lemma, because of their importance in developing the technique.

A- The Method of Arnoldi

Given a real constant unsymmetric n x n matrix A. Let v_1 a starting vector of norm one, and let m be chosen not exceeding n. The method generates a sequence of vectors $v_1, v_2, \ldots, v_m$ and an mxm upper Hessenberg matrix $H_m := V^T A V$, where $V := [v_1, \ldots, v_m]$. This is done by the following algorithm,
For $j = 1,2,\ldots,m$ do

$$h_{ij} := (A\,v_j,\ v_i),\quad i = 1,2,\ldots,j,$$

$$\hat{v}_{j+1} := A\,v_j - \sum_{i=1}^{j} h_{ij}\,v_i$$

$$h_{j+1,j} := \| \hat{v}_{j+1} \| .$$

and

$$v_{j+1} := v_{j+1} / h_{j+1,j} .$$

Define $\hat{H}_m$ as the (m+1) x m matrix whose nonzero entries are h_{ij}, then H_m is the mxm matrix obtained from $\hat{H}_m$ by deleting its last row.

B- Lemma [6]

Let V be the nxm matrix and H the mxm Hessenberg matrix generated by Arnoldi-algorithm with an initial vector $v_1 := b/\|b\|$, for some column vector b. Then the following equivalence relation,

$$(sI_n - A)^{-1} b \cong V (sI_m - H)^{-1} \|b\| e_1$$

holds, where e_1 is the first column of the mxm identity matrix.

Upon the above lemma, the proposed algorithm may then takes the following steps.

For i = 1 : p, do

Step 1. Compute the i-th starting vector $v_{1i} := \frac{b_i}{\|b_i\|}$ where b_i is the i-th column of the matrix B.

Step 2. Perform m-steps of the Arnoldi-algorithm starting with the vector v_{1i} to generate the matrices V_i and H_i.

Step 3. Solve the upper Hessenberg system,

$$(sI_m - H_i) y_i = \|b_i\| e_1.$$

Step 4. Compute the column $g_i(s)$ which is given by,

$$g_i(s) := C V_i y_i.$$

Step 5. Form your required transfer function matrix G(s) which will be given by,

$$G(s) := [g_1(s), \ldots, g_p(s)].$$

C- Remarks About The Algorithm

(1) The chosen m in the algorithm is in fact the number of output vectors.

(2) For some v_{1i} (all) the resulting system $(v_{1i}, v_{2i}, \ldots, v_{mi})$ might be not orthonormal. A remedy for this is the reorthogonalization, using the modified Gram-Schmidt method developed in [2].

(3) The method depends heavily on Arnoldi algorithm, which is stable of $O(n^2)$. For i = 1,2,...,p, generating H_i and V_i are thus requires $O(pn^2)$ operations.

(4) LU decomposition of $(sI_m - H_i)$, in step 3, requires only $\frac{1}{2} m^2$ flops. Thus, for each i, the solution of the system take m^2 flops.

(5) The computation of each column vector, in step 4, is thus requires 2mn flops.

(6) If the matrix A is large and sparse, thus if a multiprocessor computer architecture is used we can apply known [1] techniques for parallel solution of the sparse simultaneous linear systems, in step 3 (p processors are employed to compute the transfer function matrix).

III THE ALGORITHM EFFICIENCY

An efficient method was proposed to compute the transfer function matrix of linear multivariable systems. A simple operations count shows that the proposed method rquires only $m^2p + 2nmp + o(pn^2)$ Operations. This operations count include running Arnoldi algorithm p - tims , solvig p - upper Hessenberg linear systems and finally a matrix - vectors multiplications to obtain the required matrix. This count compares very favorably to the recent known exisiting methods. The Misra - Patel method [5] requires $\frac{5}{3}n^3(p+m)$ operations just to find the minimal order subsystem for each input - output pair. Moreover, it takes $\frac{1}{3}n_c^3+n_c$ operations to compute the two detrminants corresponding to each element computation of the transfer function matrix. Since, for control linear systems we have, in general, m , p < n, then our algorithm is quite efficient.

IV THE ALGORITHM ACCURACY

In Misra - Patel method [5], the accuracy was tested by computing the coefficients of the denomenator and the numerator of the transfer function matrix elements using examples with known exact coefficient transfer functions. Since our method computes the transfer function matrix ,directy, one column at a time, then we cannot compare the accuracy of the two methods. Therefore, we compare the accuracy with the exact direct method using randomly generated examples.

In Example 1, we illustrate the algorithm by computing the first column of the transfer function matrix where A is 3x3, C is 2x3, and b1 is 3x1 using s=2. We applied Arnoldi algorithm to get an upper Hessenberg matrix H of size 2x2 and the corresponding 3x2 orthonormal matrix V (using hand clculation).

In Example 2, we did the same steps, with A is 10x10, C is 6x10,and b1 is 10x1 using different values for s. In this example computations were done using the MATLAB and a software package designed by Youssif Saad(Univeristy of Illinois USA) for Arnoldi algorithm. The results are quite satisfactory.

Example 1

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{bmatrix}, \quad b_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

From Arnoldi we have

$$H = \begin{bmatrix} 1 & \sqrt{2} \\ \sqrt{2} & 2 \end{bmatrix}, \quad V = \begin{bmatrix} 1 & 0 \\ 0 & 1/\sqrt{2} \\ 0 & 1/\sqrt{2} \end{bmatrix}$$

$s = 2$

Direct Method :

We have $$(sI-A)^{-1} = \begin{bmatrix} 0 & -1/2 & 0 \\ -1/2 & 0 & 0 \\ -1/2 & -1/2 & 1 \end{bmatrix}$$

$$g_1(s) = C(sI - A)^{-1} b_1$$

$$= \begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1/2 & 0 \\ -1/2 & 0 & 0 \\ -1/2 & -1/2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -5/2 \\ -1 \end{bmatrix}$$

Proposed Method :

$$g_1(s) = C V (sI - H)^{-1} \| b_1 \| e_1 ,$$

$$= \begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1/\sqrt{2} \\ 0 & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 0 & -1/\sqrt{2} \\ -1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -5/2 \\ -1 \end{bmatrix}$$

A is 10x10 randomly generated matrix

Columns 1 through 7

$$A = \begin{bmatrix}
0.2113 & 0.5608 & 0.3076 & 0.5015 & 0.2806 & 0.4095 & 0.3874 \\
0.7560 & 0.6624 & 0.9330 & 0.4369 & 0.1280 & 0.8784 & 0.9223 \\
0.0002 & 0.7264 & 0.2146 & 0.2693 & 0.7783 & 0.1138 & 0.9488 \\
0.3303 & 0.1985 & 0.3126 & 0.6326 & 0.2119 & 0.1998 & 0.3435 \\
0.6654 & 0.5443 & 0.3616 & 0.4052 & 0.1121 & 0.5619 & 0.3760 \\
0.6284 & 0.2321 & 0.2922 & 0.9185 & 0.6857 & 0.5296 & 0.7341 \\
0.8497 & 0.2312 & 0.5664 & 0.0437 & 0.1531 & 0.6854 & 0.2616 \\
0.6857 & 0.2165 & 0.4826 & 0.4819 & 0.6971 & 0.9906 & 0.4993 \\
0.8782 & 0.8834 & 0.3322 & 0.2640 & 0.8416 & 0.5042 & 0.2639 \\
0.0684 & 0.6525 & 0.5935 & 0.4148 & 0.4062 & 0.3494 & 0.5254
\end{bmatrix}$$

Columns 8 through 10

$$\begin{bmatrix}
0.5376 & 0.5879 & 0.6489 \\
0.1200 & 0.4829 & 0.9923 \\
0.2256 & 0.2233 & 0.0500 \\
0.6274 & 0.8401 & 0.7486 \\
0.7609 & 0.1206 & 0.4104 \\
0.0496 & 0.2855 & 0.6085 \\
0.6724 & 0.8608 & 0.8544 \\
0.2017 & 0.8494 & 0.0643 \\
0.3912 & 0.5257 & 0.8279 \\
0.8300 & 0.9931 & 0.9262
\end{bmatrix}$$

H is upper Hessenberg matrix generated form Arnoldi

$$H = \begin{bmatrix}
4.0949 & 1.7610 & 0.3198 & 0.0822 & -0.2915 & -0.2681 \\
2.1100 & 0.9781 & 0.5185 & 0.0822 & -0.2341 & -0.2682 \\
0 & 0.6057 & 0.3369 & 0.2070 & -0.2759 & -0.0164 \\
0 & 0 & 0.5554 & -0.2938 & -0.1950 & -0.3541 \\
0 & 0 & 0 & 0.6514 & -0.1494 & 0.0691 \\
0 & 0 & 0 & 0 & 0.7157 & -0.4967
\end{bmatrix}$$

V is the orthonormal matrix

$$V = \begin{bmatrix}
0.3019 & 0.0010 & 0.3452 & -0.3205 & -0.2920 & -0.0253 \\
0.3043 & 0.3494 & -0.5634 & -0.4253 & -0.2839 & 0.3048 \\
0.4348 & -0.3443 & 0.0353 & -0.2925 & 0.5653 & -0.2969 \\
0.0303 & 0.5352 & 0.1208 & -0.1569 & -0.0269 & 0.0345 \\
0.2981 & -0.0514 & 0.4610 & 0.1901 & -0.4688 & 0.3966 \\
0.0666 & 0.5126 & 0.1257 & 0.0641 & 0.1550 & -0.0114 \\
0.3878 & -0.0115 & -0.2675 & 0.6903 & 0.1932 & 0.2696 \\
0.1427 & 0.3241 & 0.4581 & 0.0871 & 0.2021 & -0.1461 \\
0.2912 & 0.2492 & -0.1948 & 0.2628 & -0.4137 & -0.7439 \\
0.5267 & -0.2006 & -0.0048 & -0.1130 & 0.1439 & 0.1107
\end{bmatrix}$$

Columns 1 through 7

$$C = \begin{bmatrix} 0.2113 & 0.4524 & 0.6538 & 0.7469 & 0.1167 & 0.2260 & 0.2408 \\ 0.0824 & 0.8075 & 0.4899 & 0.0378 & 0.6250 & 0.8159 & 0.6907 \\ 0.7599 & 0.4832 & 0.7741 & 0.4237 & 0.5510 & 0.2284 & 0.1062 \\ 0.0087 & 0.6135 & 0.9626 & 0.2613 & 0.3550 & 0.8553 & 0.2640 \\ 0.8096 & 0.2749 & 0.9933 & 0.2403 & 0.4943 & 0.0621 & 0.7034 \\ 0.8474 & 0.8807 & 0.8360 & 0.3405 & 0.0365 & 0.7075 & 0.4021 \end{bmatrix}$$

Columns 8 through 10

$$\begin{bmatrix} 0.6553 & 0.9166 & 0.0503 \\ 0.9700 & 0.1402 & 0.5782 \\ 0.0380 & 0.7054 & 0.2432 \\ 0.0988 & 0.0178 & 0.9448 \\ 0.2560 & 0.2611 & 0.5876 \\ 0.5598 & 0.1358 & 0.7256 \end{bmatrix}$$

$$b1 = \begin{bmatrix} 0.5667 \\ 0.5712 \\ 0.8160 \\ 0.0569 \\ 0.5596 \\ 0.1249 \\ 0.7279 \\ 0.2678 \\ 0.5465 \\ 0.9885 \end{bmatrix}$$

```
» s1=10

» g1=C*inv(s1*i-A)*b1

g1 =

    0.4258
    0.5666
    0.4691
    0.5007
    0.5465
    0.6026
```

```
» g1=C*V*inv(s1*I-H)*norm(b1)*e1

g1 =

    0.4255
    0.5662
    0.4692
    0.5009
    0.5464
    0.6026
```

```
» s2=20

» g1=C*inv(s2*i-A)*b1

g1 =

    0.1357
    0.1865
    0.1584
    0.1708
    0.1917
    0.2011
```

```
» g1=C*V*inv(s2*I-H)*norm(b1)*e1

g1 =

    0.1357
    0.1865
    0.1584
    0.1708
    0.1917
    0.2011
```

```
» s3=30
» g1=C*inv(s3*i-A)*b1
g1 =
    0.0802
    0.1115
    0.0955
    0.1033
    0.1171
    0.1208
```

```
» g1=C*V*inv(s3*I-H)*norm(b1)*e1
g1 =
    0.0802
    0.1115
    0.0955
    0.1033
    0.1171
    0.1208
```

```
» s4=40
» g1=C*inv(s4*i-A)*b1
g1 =
    0.0569
    0.0795
    0.0684
    0.0741
    0.0843
    0.0863
```

```
» g1=C*V*inv(s4*I-H)*norm(b1)*e1
g1 =
    0.0569
    0.0795
    0.0684
    0.0741
    0.0843
    0.0863
```

```
s5 = 50
» g1=C*inv(s5*i-A)*b1
g1 =
    0.0441
    0.0618
    0.0533
    0.0577
    0.0659
    0.0672
```

```
» g1=C*V*inv(s5*I-H)*norm(b1)*e1
g1 =
    0.0441
    0.0618
    0.0533
    0.0578
    0.0659
    0.0672
```

V- CONCLUSION

An Arnoldi-based method is presented for efficient computation of transfer function matrices one column at a time. The efficiency and the performance (in case of large and sparse state matrices) were compared with some recent methods. An illustrative numerical example was given to test the accuracy compared with the exact direct method.

REFERENCES

1) Calahan, D.A., "Parallel Solution of Sparse Simultaneous Linear Equations", Proc. 11th Ann. Allerton Conf. Circuit and syst. theory, pp. 729 (1973).

2) Daniel, G.W., Gragg, W.B., Kaufman, L., and Stewart, G.W., "Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization", Math. Comp. 30, 772-795 (1976).

3) Emami - Naeini, A. and VanDooren, P., "On computation of transmission Zeros and transfer functions". Proc. IEEE Control and Dec. Conf. orlando, pp. 51-56 (1982).

4) Galub, G.H., and Van Loan, C., "Matrix Computations". The John Hopkins University, (1983).

5) Misra, P. and Patel, R.V., "Computation of transfer function matrices of linear multivariable systems", Automatica, Vol.23, No.5, 635-640 (1987).

6) Nasr, H. and Ramadan, M., "An efficient frequency response computation using Arnold's algorithm", Proc. 16th International Conference, Ain Shams University Cairo, 259-279 (1991).

7) Verga, A. and Sima, V., "Numerically stable algorithm for transfer function matrix evaluation", Int. J. Control, 33, 1123-1133 (1981).

8) Wilkinson, J.H., "The Algebraic Eigenvalue Problem", Oxford University Press, Oxford (1965).