Abstract: - In this paper one presents an algorithm for project of a reduced order observer (ALGLOOR) with applications to flying objects’ move. The command law is an optimal one in rapport with state vector of the observer. Numeric calculus examples are also presented: for longitudinal and lateral move and vertical move of a rocket in vertical plane in rapport with equal signal line; using Matlab/Simulink model of closed loop system one has made numeric calculus programs and with them one obtained different time characteristics, which expresses state variables’ dynamics of the flying objects dynamic models.

Key-Words: - algorithm, optimal, control law, rocket, state vector, reduced order observer, command

1 Introduction

Observers are used for estimation of flying object state vector using its input and output signals. Measuring some of the state vector’s components (output vector’s components) one reduces sensors’ number especially the number of sensors for reachless or difficult measurable state variables (e.g. elastic deformations) [1], [2], [3]. First observer structures have been developed by Luenbergher in 1966, 1971, 1979. The observers are used both for systems with known inputs and unknown inputs systems; their project belongs to different researchers (Bhatta-Charyya, 1978; Chen and Patton, 1999; Darouach, 1994; Hostetter and Meditch, 1973; Hou and Müller, 1992; Hui and Zak, 1993 and 2005; Kudva, 1980; Kurek, 1983; Wang, 1975; Yang and Wilde, 1988; Krzminski and Kakzorek, 2005 and so on). Recent observers are obtained for systems with some unknown inputs or for systems with subsystems whose dynamic is unknown. For such systems one may use reduced order observers. Unknown inputs may be disturbances, errors and so on.

The estimator (filter) Kalman – Bucy [4], [5], [6] is the best observer for stochastic systems’ state estimation. In case of nonlinear systems with partial known inputs or with unknown subsystems adaptive observers have been made; these observers use neural networks [7], [8], [9], [10], [11].

Recent researches deal with project of observers for the intern delay systems [12], [13], [14], based on Liapunov theory; observers’ project means matrices inequalities solve. Linear state observers are used for $x$ state estimation for linear system using measured variables vector $y$, the input vector or some components of this vector.
2 Dynamic models of flying objects and reduced order observer

Flying objects’ dynamic may be described by equations [15]
\[ \dot{x} = Ax + Bu + Eu_p, \]  
\[ y = Cx, \]  
where \( x \) is the state vector \((n \times 1)\) of aircraft model, \( y \) – output vector \((p \times 1)\), \( u \) – the vector containing knowable inputs \((m_1 \times 1)\), \( u_p \) – vector \((m_2 \times 1)\) of unknown inputs (disturbances or nonlinear functions of non modeled systems) [16], \( A \) – matrix \((n \times n)\), \( B \) – matrix \((n \times m_1)\), \( E \) – matrix \((n \times m_2)\), \( C \) – matrix \((p \times n)\); matrices \( A, B, C \) are known.

Let’s consider the reduced order observer described by equations [17], [18]
\[ M_2 = Fz + Gu + Hy, \]  
\[ \dot{\hat{x}} = Pz + Qy, \]  
with \( z(r \times 1), \dot{\hat{x}}(n \times 1), M(r \times r), F(r \times r), \) det \( M = 0 \) and \( \text{rang} \ M \leq n-q, G(r \times m), H(r \times p), P(n \times r), \) \( Q(n \times p); m = m_1 + m_2; e \) is the observer’s error
\[ e = z - Nx, \]  
By derivation of equation (5), replacing \( \dot{\hat{x}} \) with form (1), \( y \) with form (2), \( \dot{x} \) with form (3) and assenting that all coefficients of \( x, u \) and \( u_p \) be null,
\[ G = MNB, \]  
\[ HC = MNA - FN, \]  
\[ MNE = 0, \]  
one obtains the equation of error \( e \)
\[ M\dot{e} = Fe. \]  
Error \( (\dot{\hat{x}} - x) \) may be expressed in rapport with \( e \).

Taking into account equations (4) and (5), one results
\[ \dot{x} - x = Pe \]  
if
\[ PN + QC = I \Leftrightarrow [P \ Q][N \ C]^T = I. \]  
\( I \) – unity matrix.

3 Project algorithm of the reduced order observer (ALGLOOR)

First of all one chooses matrix \( N \) of form
\[ N = \begin{bmatrix} N_1 & N_2 \\ N_3 & N_4 \end{bmatrix} = \begin{bmatrix} I_{(n-q)(n-p)} & 0_{(n-q)p} \\ 0_{(n-p)(n-q)} & I_{(n-p)p} \end{bmatrix}, \]  
where \( I_{(n-q)(n-p)} = I_{n-p} \) is unity matrix and \( I_{(n-p)p} \) is a matrix whose elements are equal with 1 at intersection of line \( i \) and column \( j = i \) and all the other elements are null; \( p = \text{rang} \ C, q = \text{rang} \ E = n - p \).

One calculates matrix \( M \) so that condition (8) be fulfilled. For this, matrix \( M \) is partitioned
\[ M = \begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix}, \]  
\( M_1 = \begin{bmatrix} M_{1(n-p)(n-q)} & M_{2(n-p)(n-q)} \\ M_{3(n-q)(n-p)} & M_{4(n-q)(n-p)} \end{bmatrix}. \]  
With \( E \) of form
\[ E = \begin{bmatrix} E_1 \\ E_2 \end{bmatrix}, \]  
\[ E_1 = \begin{bmatrix} (n-p) \times q \\ (n-q) \times p \end{bmatrix}, \]  
equation (8) is equivalent with system
\[ M_2 E_1 + M_1 E_2 = 0, \]  
\[ M_4 E_1 + M_3 E_2 = 0. \]  
Choosing
\[ M_2 = 0_{(n-p)(n-p)}, M_4 = 0_{(n-q)(n-p)}, \]  
one results
\[ M = \begin{bmatrix} M_1 \\ M_3 \end{bmatrix}, \]  
\[ M_1 E_2 = 0, M_3 E_2 = 0. \]  
With \( M \) and \( N \) calculated, one determines matrix \( G \) using equation (6). Also, one calculates matrices \( F \) and \( H \) with (7). For this, matrix \( F \) is partitioned as follows
\[ F = \begin{bmatrix} F_{1(n-p)(n-q)} & 0_{(n-p)(n-p)} \\ F_{2(n-q)(n-q)} & 0_{(n-q)(n-p)} \end{bmatrix}. \]  
With \( H \) and \( A \) of forms
\[ H = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}, \]  
\[ H_1 = \begin{bmatrix} (n-p) \times p \\ (n-q) \times p \end{bmatrix}, \]  
\[ A = \begin{bmatrix} A_{1(n-p)(n-p)} & A_{2(n-p)(n-p)} \\ A_{3(n-p)(n-q)} & A_{4(n-q)(n-p)} \end{bmatrix}, \]  
condition (7) is equivalent with system
\[ M_1 A_1 = H_1 C_1, \]  
\[ M_1 A_2 - F_1 = H_2 C_1, \]  
\[ M_3 A_3 = H_2 C_1, \]  
\[ M_3 A_4 - F_3 = H_2 C_2; \]  
one results
\[ H_1 = M_1 A_1 C_1^*, \]  
\[ H_2 = M_3 A_3 C_1^*, \]  
\[ F_1 = M_1 A_2 - M_3 A_3 C_1^* C_2, \]  
\[ F_3 = M_3 A_4 - M_3 A_3 C_1^* C_2. \]  
One yield
\[ H = \begin{bmatrix} M_1 A_1 C_1^* \\ M_3 A_3 C_1^* \end{bmatrix}, \]  
\[ F = \begin{bmatrix} M_1 A_2 - M_3 A_3 C_1^* C_2 & 0_{(n-p)(n-p)} \\ M_3 A_4 - M_3 A_3 C_1^* C_2 & 0_{(n-q)(n-p)} \end{bmatrix}. \]  
Matrices \( P \) and \( Q \) are obtained from equation (11). These matrices are partitioned as follows
(equations (3) and (4)) is presented.

One remarks that the algorithm ALGLOOR doesn’t use non singular transformations like other algorithms [19], [20], [21].

4 Numeric examples

Example 4.1 (aircraft longitudinal move)

A reduced order observer for state \( \hat{x} \) estimation described by equation (31) from [1] for longitudinal move with

\[
C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},
\]

\( n = 4, \ p = 2, \ q = 2 \)

may be projected using ALGLOOR algorithm

\[
\left[ \begin{array}{c} \Delta V_x \\ \Delta \alpha \\ \Delta \delta \\ \Delta \omega \end{array} \right] = \begin{bmatrix} -0.007 & 0.012 & -9.81 & 0 \\ -0.128 & -0.54 & 0 & 1 \\ 0.065 & 0.96 & 0 & -0.99 \end{bmatrix} \begin{bmatrix} \Delta V_x \\ \Delta \alpha \\ \Delta \delta \\ \Delta \omega \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},
\]

\( (31) \)

Matlab program utilized is the first presented in Section 4: one obtained matrices

\[
N = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},
\]

\( M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},
\]

\( G = \begin{bmatrix} 0 \\ -12.5 \\ 0 \\ -12.5 \end{bmatrix},
\]

\( (32) \)

\[
H = \begin{bmatrix} 0 \\ 0.065 & 0.065 \\ 0 \\ 0.065 & 0.065 \end{bmatrix},
\]

\( F = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix},
\]

\( (32) \)

\[
P = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix},
\]

\( Q = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},
\]

\( (32) \)

In fig.2 Matlab/Simulink model of the closed loop system

In fig.2 Matlab/Simulink model of the closed loop system described by equations (1)-(4) and \( u = -K \hat{x} \) is presented. For \( u = 0 \) (open loop) one obtains characteristics \( x(t) \) (blue) and \( \hat{x}(t) \) (red)-fig.3. These characteristics are also obtained for closed loop system – fig.4.
ALGLOOR may be used with good results using longitudinal move’s model from [22]

**Example 4.2 (aircraft lateral move)**

Similar, for lateral move described by equation [1],

\[
\begin{bmatrix}
\Delta \beta \\
\Delta \phi \\
\Delta \omega$
\end{bmatrix}
= \begin{bmatrix}
0.998 \\
0.005 \\
0$
\end{bmatrix}
+ \begin{bmatrix}
-0.0558 \\
0.385 \\
0$
\end{bmatrix} \Delta \phi
\]

are known

\[
C = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix},
E^r = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}
\]

\[
n = 4, p = q = 2;
\]

one obtains

\[
N = \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix},
M = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix},
G = \begin{bmatrix}
0.15 & 1.06 \\
0 & 0 \\
0.15 & 1.06 \\
0 & 0
\end{bmatrix},
\]

\[
H = \begin{bmatrix}
0.30 & 0.38 \\
0 & 0.08 \\
0.30 & 0.38 \\
0 & 0.08
\end{bmatrix},
F = \begin{bmatrix}
-0.46 & 0 & 0 & 0 \\
-0.46 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix},
\]

\[
P = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix},
Q = \begin{bmatrix}
1 & 0 \\
0 & 1 \\
0 & 0 \\
0 & 0
\end{bmatrix}
\]

\[
\dot{\chi}(t) = M \chi(t) + N \Delta \phi + G \Delta \phi + H \delta + F \beta + P \omega + Q \theta.
\]

Characteristics \( \chi(t) \) (blue) and \( \hat{\chi}(t) \) (red) are presented in fig.5 (without controller) and fig.6 (with controller \( u = -K \hat{\chi} \)).

**Example 4.3 (rocket move)**

One considers the case of a rocket move in vertical plane; it is directed by three points method (the colinear point CP–R–T; CP – control point, R – rocket, T - target). Lateral deviation \( z \) in rapport with equal signal line is described by differential equation [23], [24]

\[
\dot{z} = V\theta + Vf_T,
\]

where \( V \) is the flight velocity, \( \theta \) – slope of the trajectory \( \theta = \theta - \alpha, \theta \) – pitch angle, \( f_T \) – disturbance. \( \theta \) is defined by [24]

\[
T_r \dot{\theta} = \alpha - T_r \frac{g}{V} \cos \theta.
\]

Expressing the above equation in variables \( \alpha \) and \( \theta \), it becomes

\[
T_r \dot{\theta} = T_r \dot{\alpha} + \alpha - T_r \frac{g}{V} \cos \theta
\]

or has the liniarised form
where \( d_1, d_2, d_3, d_4 \) are read using diagrams or graphic characteristics for different rocket types at different flight seconds. For instance, for an Oerlikon rocket, in the 10th flight second
\[
d_1 = 1.5 \text{ s}^{-1}, d_2 = 40 \text{ s}^{-2}, d_3 = -0.1 \text{ s}^{-2}, d_4 = 1.2 \text{ s}^{-2},
\]
\[
V = 400 \text{ m/s}, T_v = 0.66 \text{ s}, \omega_0 = 4.669 \text{ s}^{-1}, \xi = 0.062,
\]
\[
k_\theta = d_1 = 40 \text{ s}^{-2}, a_1 = 0.924 \text{ s}^{-1}, a_0 = 23.18 \text{ s}^{-2}.
\]
For the 40th flight second
\[
d_1 = 2.5 \text{ s}^{-1}, d_2 = 100 \text{ s}^{-2}, d_3 = -80 \text{ s}^{-2}, d_4 = 1 \text{ s}^{-1},
\]
\[
V = 400 \text{ m/s}, T_v = 0.4 \text{ s}, \omega_0 = 0.983 \text{ s}^{-1}, \xi = 0.0212,
\]
\[
k_\theta = d_1 = 100 \text{ s}^{-2}, a_1 = 2.115 \text{ s}^{-1}, a_0 = 87.7881 \text{ s}^{-2}.
\]
For the 50th flight second
\[
d_1 = 1 \text{ s}^{-1}, d_2 = 25 \text{ s}^{-2}, d_3 = -60 \text{ s}^{-2}, d_4 = 0.4 \text{ s}^{-1},
\]
\[
V = 400 \text{ m/s}, T_v = 1 \text{ s}, \omega_0 = 7.77 \text{ s}^{-1}, \xi = 0.011,
\]
\[
k_\theta = d_1 = 25 \text{ s}^{-2}, a_1 = 0.8197 \text{ s}^{-1}, a_0 = 61.2082 \text{ s}^{-2}.
\]
Choosing the state vector \( x^T = [\xi \ \dot{\xi} \ \Delta \alpha \ \dot{\Delta} \ \dot{\alpha} \ \dot{\theta}] \), system formed by equations (38), (45) and (46) becomes
\[
\begin{align*}
\dot{x}_1 &= x_2, \\
\dot{x}_2 &= \frac{V}{T_v} x_3 + W_f, \\
\dot{x}_3 &= -\frac{1}{T_v} x_1 + x_4 + \frac{1}{T_v} \ddot{\alpha}, \\
\dot{x}_4 &= -a_0 x_1 - a_3 x_3 + k_\delta.
\end{align*}
\]
If the input is \( u = \delta \) and the disturbances vector is \( u^T = [\tilde{\alpha} \ W_f] \), the above equations system has the form
\[
\dot{x} = Ax + Bu + Eu_r,
\]
where
\[
A = \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & V & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & -a_0 & -a_3
\end{bmatrix} ,
B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ k_\delta \end{bmatrix} ,
E = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}
\]
Using ALGLOOR for the 10th flight second, one obtains
\[
C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} ,
N = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}
\]
\[
M = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} ,
G = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 600 & 0 \end{bmatrix},
H = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 600 & 0 \end{bmatrix}
\]
\[
F = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} ,
P = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} ,
Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\]
Eigen values of the open loop system are
\[ s_1 = 0, s_2 = 0, \]
\[ s_3 = 4.82, s_4 = -4.52. \] (55)

In closed loop system one obtains the characteristics from fig.7.

Similarly, for the 40th and 50th flight second, it results

\[
\begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}, \quad \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

respectively,

\[
\begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}, \quad \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

and characteristics from fig.8 and fig.9.

5 Matlab programs

One presents the program for the longitudinal move of the flying object and for rocket move in vertical plane at 10th flight second. The programs for lateral move or for rocket move in vertical plane at 40th and 50th flight second are similar. In programs presented in Subsections 5.1 and 5.2 one runs other subprograms: ALGLX & ALGLX10. These programs are shown in Subsections 5.3 and 5.4.

5.1 ALGLOOR program (longitudinal move)
clear all; run ALGLX; close all;
clear C; clear contor; clear e; clear i;
clear r; clear m;
clear P; clear Q; KK=K;
C=[1 0 0 0;0 1 0 0];
E=[1 0; 0 1; 0 0; 0 0];
n=size(A,1); m=size(B,2);
q=size(E,2); p=size(C,1);
x0=[100; 1; 0; 10];
xc0=[10; 0; -10; 20];
up=[-0.0375; -0.0963];
deltap=[5];
A1=A(1:(n-p),1:(n-p));
A2=A(1:(n-p),(n-p+1):size(A,2));
A3=A((n-p+1):size(A,1),1:(n-p));
A4=A((n-p+1):size(A,1),(n-p+1):size(A,2));
C1=C(:,1:(n-p)); C2=C(:,(n-p+1):n);

% Step 1
N1=zeros((n-q),(n-p));
N2=eye((n-q),p);
N3=eye((n-p),(n-p));
N4=zeros((n-p),p);
N=[N1 N2; N3 N4];

% Step 2
M1=zeros((n-p),(n-q));
M2=zeros((n-p),(n-p));
M3=eye((n-q),(n-q));
M4=zeros((n-q),(n-p));
M=[M1 M2; M3 M4];

% Step 3
I1=eye((n-p),(n-p));
I2=zeros((n-p),(n-n));
I3=zeros((n-q),(n-n));
I4=eye((n-q),(n-q));
I=[I1 I2; I3 I4];
P1=zeros((n-p),(n-n));
P2=zeros((n-p),(n-n));
P3=eye((n-q),(n-q));
P4=pinv(c2)*c1;
P=[P1 P2; P3 P4];
Q1=eye((n-p),p);
Q2=zeros((n-q),p);
Q=[Q1 Q2];

% Step 4
F1=M1*A4-M1*A3*pinv(C1)*C2;
F2=zeros((n-p),(n-n));
F3=M3*A4-M3*A3*pinv(C2)*C1;
F4=zeros((n-q),(n-n));
F=[F1 F2; F3 F4];
H1=M1*A3*pinv(C1);
H2=M3*A3*pinv(C1);
H=[H1 H2];

% Step 5
G=M*N*B; s=rand(1)+randn(1)*i;
MAT=[s*eye(n-A;E;C zeros(2,2)];
rank(MAT)
if rank(MAT)~=n+q
disp('The method isn’t useful');
end
if M*N*E==zeros(n-p),q; zeros((n-q),q)
disp('First convergence condition isn’t fulfilled');
end
if M*N*B-G==zeros(n,m)
disp('Second convergence condition isn’t fulfilled');
end
if M*N*A-F*N==H*C
disp('Third convergence condition isn’t fulfilled');
end
if P*N+Q*C==I
disp('Fourth convergence condition isn’t fulfilled');
end

5.2 ALGLOOR program (rocket move)

% Rocket move s=10
run ALGX10;
close all; clear C; clear contor; clear e;
clear P1; clear P2; clear P3; clear P4;
clear P; clear P; clear k1;
KK=K;

% Calculus data
n=size(A,1); m=size(B,2);
q=size(E,2); p=size(C,1);
x0=[10; 0; 1; 100];
xc0=[10; 1; 0; 100];
up=randn(2,1);

% Matrices A and C partition
A1=A(1:(n-p),1:(n-p));
A2=A(1:(n-p),(n-p+1):size(A,2));
A3=A((n-p+1):size(A,1),1:(n-p));
A4=A((n-p+1):size(A,1),(n-p+1):size(A,2));
C1=C(:,1:(n-p)); C2=C(:,(n-p+1):n);

% Step 1
N1=zeros((n-q),(n-p));
N2=eye((n-q),p);
N3=eye((n-p),(n-p));
N4=zeros((n-p),p);
N=[N1 N2; N3 N4];

% Step 2
M1=zeros((n-p),(n-q));
M2=zeros((n-p),(n-p));
M3=eye((n-q),(n-q));
M4=zeros((n-q),(n-p));
M=[M1 M2; M3 M4];

% Step 3
I1=eye((n-p),(n-p));
I2=zeros((n-p),(n-q));
I3=zeros((n-q),(n-p));
I4=eye((n-q),(n-q));
I=[I1 I2; I3 I4];
P1=zeros((n-p),(n-n));
P2=zeros((n-p),(n-n));
P3=eye((n-q),(n-q));
P4=pinv(c2)*c1;
P=[P1 P2; P3 P4];
Q1=eye((n-p),p);
Q2=zeros((n-q),p);
Q=[Q1 Q2];

% Step 4
F1=M1*A4-M1*A3*pinv(C1)*C2;
F2=zeros((n-p),(n-n));
F3=M3*A4-M3*A3*pinv(C2)*C1;
F4=zeros((n-q),(n-n));
F=[F1 F2; F3 F4];
H1=M1*A3*pinv(C1);
H2=M3*A3*pinv(C1);
H=[H1 H2];

5.2 ALGLOOR program (rocket move)
M3=zeros((n-q),(n-q));
M4=eye((n-q),(n-p));
M=[M1 M2;M3 M4];

% Step 3
I1=eye((n-p),(n-p));
I2=zeros((n-p),(n-q));
I3=zeros((n-q),(n-p));
I4=eye((n-q),(n-q));
I=[I1 I2;I3 I4];
P1=zeros((n-p),(n-q));
P2=eye((n-p),(n-p));
P3=zeros((n-q),(n-p));
P4=-(pinv(C2))*C1;
P=[P1 P2;P3 P4];
Q1=zeros((n-p),p);
Q2=pinv(C2);
Q=[Q1;Q2];

% Step 4
F1=zeros((n-p),(n-q));F2=A1;
F3=zeros((n-q),(n-q));F4=A1;
F=[F1 F2;F3 F4];
H1=A2;H2=A2;
H=[H1;H2];

% Step 5
G=M*N*B;

% Tests
s=rand(1)+randn(1)*i;
MAT=[s*eye(n)-A E;C zeros(2,2)];
if rank(MAT)~=n+q
    disp ('The method isn’t useful');
end
if M*N*E~=[zeros((n-p),q);zeros((n-q),q)]
    disp ('First convergence condition isn’t fulfilled');
end
if M*N*B-G~=[zeros(n,m)]
    disp ('Second convergence condition isn’t fulfilled');
end
if M*N*A-F*N~=[zeros(n,m)]
    disp ('Third convergence condition isn’t fulfilled');
end
if P*N+Q*C~=I
    disp ('Fourth convergence condition isn’t fulfilled');
end

% K=0 (without controller)
K=zeros(1,n); sim('schrach');
subplot(221);plot(t,x1,'b',t,xc1,'r--');grid;
subplot(222);plot(t,x2,'b',t,xc2,'r--');grid;
subplot(223);plot(t,x3,'b',t,xc3,'r--');grid;
subplot(224);plot(t,x4,'b',t,xc4,'r--');grid;

% K=K (with controller)
K=KK;
h=figure;
sim('schrach');
subplot(221);plot(t,x1,'b',t,xc1,'r--');grid;
subplot(222);plot(t,x2,'b',t,xc2,'r--');grid;
subplot(223);plot(t,x3,'b',t,xc3,'r--');grid;
subplot(224);plot(t,x4,'b',t,xc4,'r--');grid;

5.3 ALGLX program

close all;
A=[-0.007 0.012 -9.81 0; -0.128 -0.54 0 1; 0 0 0 1; 0.065 0.96 0 -0.99];
B=[0;0.04;0;-12.5];
Q=[10 0 0 0 10 0 0 0 100 0; 0 0 0 0 0 0 1]; R=[2];
[K,P,E] = LQR(A,B,Q,R);
I2=[1 0; 0 1];
N3=randn(4,3);
contor=1;
T(:,1)=B(:,1);
for i=1:4
    for j=1:3
        T(i,j+1)=N3(i,j);
    end
end

Ab=(inv(T))*A*T;
Bb=(inv(T)*B);
Kb=place(Ab,Bb,E);
e=eig(Ab-Bb*Kb);
k1=Kb(1);k21=Kb(2);
k22=Kb(3);k23=Kb(4);
r1=5;Rb=[r1];R=Rb;
Pb=r1*[k1 k21 k22 k23; k21 10 0; k22 0 1; k23 0 0 1];

ee=eig(Rb);
Qb=-(Pb*Ab+(transpose(Ab))*Pb-Pb*Bb*Kb);
PPK=transpose(inv(T))*Pb*inv(T);
KK=inv(Rb)*transpose(B)*PPK;
EEE=eig(Ab*KK);
m=rank(T);
while real(EEE(1))>0 | real(EEE(2))>0 | real(EEE(3))>0 | real(EEE(4))>0 | m<4
    N3=randn(4,3);
    contor=contor+1;
    T(:,1)=B(:,1);
    for i=1:4
        for j=1:3
            T(i,j+1)=N3(i,j);
        end
    end
    Ab=(inv(T))*A*T;
    Bb=(inv(T)*B);
Kb=place(Ab,Bb,E);
e=eig(Ab-Bb*Kb);
k1=Kb(1);k21=Kb(2);
k22=Kb(3);k23=Kb(4);
r1=1;Rb=[r1];R=Rb;
Pb=r1*[k1 k21 k22 k23;k21 10 0;k22 0 10;k23 0 0 1];
ee=eig(Rb);
Qb=-(Pb*Ab+(transpose(Ab))*Pb-Pb*Bb*Kb);
PPP=transpose(inv(T))*Pb*inv(T);
KKK=inv(R)*transpose(B)*PPP;
EEE=eig(A-B*KKK);
m=rank(T);
end
Q=transpose(inv(T))*Qb*inv(T); R=Rb;
[KK,PP,EE]=LQR(A,B,Q,R);

5.4 ALGLX10 program
close all;
clear all;
d1=1.5;d2=40;d3=-20;d4=1.2;V=400;T1=1/d1;
Tv=T1; w=sqrt(d1*d4-d3);csi=(d1+d4)/(2*w);
kd=d2; a0=w0^2-(2*csi*w0)/Tv+1/(Tv^2);
a1=2*csi*w0-1/Tv;k0=70;
A=[0 1 0 0;0 0 V/Tv 0;0 0 -1/Tv 1;0 0 -a0 -a1];
B=[0;0;0;kd];
C=[0 0 1 0 0 0 0 1];
E=[0 0 0;1 0 0 1];
Q=[10 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1];R=[2];
[K,P,E]=LQR(A,B,Q,R); I2=[1 0 0 1];

% Matrix T definition
N3=randn(4,3);
contor=1; T(:,1)=B(:,1);
for i=1:4
  for j=1:3
    T(i,j+1)=N3(i,j);
  end
end

% Variant 1
Q=transpose(inv(T))*Qb*inv(T); R=Rb;

6 Conclusion
A new algorithm for project of a reduced order observer is presented; it may be used in different applications and in case of aircrafts’ automat command system. Theoretical results are validated by numerical simulations using calculus program made by authors using dynamic models of aircrafts, rockets’ movement equations and optimal control laws.

References:


