Analysis and Control of the Lorenz System

Nathan Isaman
Aeronautical and Astronautical Engineering
University of Washington
Seattle, Washington 98105
Email: nisaman@uw.edu

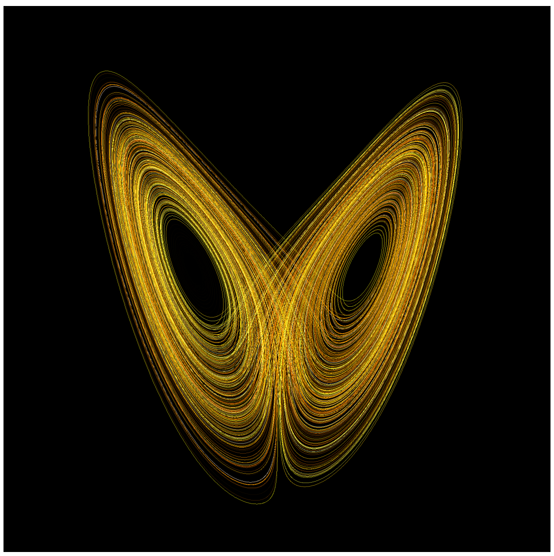


Figure 1. Drift for a Lorenz System [1]

1. Introduction

The system investigated in this paper is the Lorenz System; a popular nonlinear system due to its chaotic properties and aesthetically pleasing limit cycle, as seen in Figure 1. The Lorenz system is present in several simplified models for physical systems. A few examples of these systems include: lasers, dynamos, brushless DC motors, electric circuits, and chemical reactions [1]. Each of these systems could benefit from control so the analysis of the system and its controllability and observability are worthwhile endeavours.

2. Model

The basic model for the uncontrolled Lorenz system consists of the following system of nonlinear differential equations.

$$\begin{gathered}\dot{x}_1 = \sigma(x_2 - x_1)\\ \dot{x}_2 = x_1(\rho - x_3) - x_2\\ \dot{x}_3 = x_1 x_2 - \beta x_3\end{gathered}$$

When some control elements are added to this system, the system of equations becomes the following.

$$\begin{gathered}\dot{x}_1 = \sigma(x_2 - x_1)\\ \dot{x}_2 = x_1(\rho - x_3) - x_2 + u_1\\ \dot{x}_3 = x_1 x_2 - \beta x_3 + u_2\end{gathered}$$

This controlled Lorenz system can be further modified so that is in the desirable control affine form:

$$\begin{gathered}\dot{x} = f_0(x) + \sum f_i(x) u_i(x)\\ y(x) = h(x)\end{gathered}$$

Control affine form for the controlled Lorenz system assuming full state output:

$$\dot{x} = \begin{bmatrix}\sigma(x_2 - x_1)\\ x_1(\rho - x_3) - x_2\\ x_1 x_2 - \beta x_3\end{bmatrix} + \begin{bmatrix}0 & 0\\ 1 & 0\\ 0 & 1\end{bmatrix} u$$

$$y(x) = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} x$$

3. Analysis

The Lorenz system's controllability, observability, feedback linearizability, and control implementation will be inspected and discussed in this section.

3.1. Controllability

The controllability of the Lorenz system can be determined using the system ideal, $\mathcal{I}$. For this system, the ideal is composed of the following vector fields.

$$\mathcal{I} = span\{f_1, f_2, [f_0, f_1], [f_0, f_2], \ldots\}$$

$$= span\left\{\begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}-\sigma\\1\\-x_1\end{bmatrix}, \begin{bmatrix}0\\x_1\\\beta\end{bmatrix}, \ldots\right\}$$

Since $\sigma \in \mathbb{R}$ and $\sigma \neq 0$, the first three terms of the above ideal span $\mathbb{R}^3$. This implies the drift is in the span of the ideal for all states and the reachable set is all of $\mathbb{R}^3$. This means the Lorenz system under investigation is Small Time Locally Controllable.

3.2. Observability

The local observability of a system can be determined by evaluating the rank of the co-distribution of either the Little or Big Observability algebra. The Little Observability Algebra's co-distribution was first investigated to determine this system's local observability.

$$\begin{aligned}\mathcal{O}_1 &= span\{h_1, h_2, h_3, [f_0, h_1], [f_0, h_2], \dots\} \\ &= span\{ \begin{bmatrix} x_1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ x_2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ x_3 \end{bmatrix}, \dots \}\end{aligned}$$

thus

$$d\mathcal{O}_1 = span\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \dots \}$$

From the above, we can see that the observability co-distribution, $d\mathcal{O}_1$, is full rank for all states. This means the Lorenz system is locally observable for all states around x_0 with the current full-state observer. The full-state observer makes the discussion of observability somewhat trivial, thus reduced levels of observation were investigated. This investigation also turned out to be beneficial for efforts in Sections 3.3 and 3.4. Each observer, $h_1, h_2,$ and h_3, was tested on their own to determine if a single observer could be used and still maintain local observability. Single observer cases did not result in full rank observability co-distributions. From there, pairs of observers were investigated. A combination of h_3 and either h_1 or h_2 was required for a full rank $d\mathcal{O}_i$ and thus local observability around x_0. For the sake of keeping this paper tidy, the various co-distributions will be omitted from this section, but can be found in the MATLAB output attached.

3.3. Feedback Linearization

The system's feedback linearizability was determined using the following theorem:

Theorem: Suppose $[f_1(x_0), \dots, f_m(x_0)]$ has rank m. Then the SSELP is solvable if and only if:

1. for each $0 \leq i \leq n-1, \Delta_i$ has constant dimension near x_0.
2. Δ_{n-1} has dimension n.
3. for each $0 \leq i \leq n-2, \Delta_i$ is involutive.

For the Lorenz system under investigation, the distributions required to verify the conditions in the above theorem are the following:

$$\begin{aligned}&\Delta_0 = span\{f_1, f_2\} \\ &\Delta_1 = span\{\Delta_0, [f_0, f_1], [f_0, f_2]\} \\ &\Delta_2 = span\{\Delta_1, [f_0, [f_0, f_1]], [f_0, [f_0, f_2]]\}\end{aligned}$$

These distributions, when evaluated at $x_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$, are the following:

$$\begin{aligned}&\Delta_0 = span\{ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \} \\ &\Delta_1 = span\{\Delta_0, \begin{bmatrix} -\sigma \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ \beta \end{bmatrix} \} \\ &\Delta_2 = span\{\Delta_1, \begin{bmatrix} -\sigma^2 - \sigma \\ \rho\sigma + 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ \beta^2 \end{bmatrix} \}\end{aligned}$$

Criteria 1 and 2 are satisfied from the above theorem since $\Delta_0, \Delta_1,$ and Δ_2 have constant dimension near x_0 and Δ_2 has dimension equal to three, which is the dimension of the state manifold. Next, the involutivity of Δ_0 and Δ_1 were investigated.

Δ_0 :

$$[f_1, f_2] = 0 \implies \Delta_0 \text{ involutive}$$

Δ_1 :

$$\begin{aligned}&[f1, [f_0, f_1]] = 0 \\ &[f1, [f_0, f_2]] = 0 \\ &[f2, [f_0, f_1]] = 0 \\ &[f2, [f_0, f_2]] = 0 \\ &\implies \Delta_1 \text{ involutive}\end{aligned}$$

Since Δ_0 and Δ_1 are involutive, the system satisfies all three of the theorem's criteria. The SSELP is therefore solvable for this system.

The output functions h_1 and h_3 will be used in order to reduce the complexity of the forthcoming feedback linearization. The relative degree vector was then calculated using these output functions and taking successive derivatives of $y(x)$ to see when the inputs show up in each term.

$$\begin{aligned}&y(x) = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} \qquad \dot{y}(x) = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} \sigma(x_2 - x_1) \\ x_1x_2 - \beta x_3 + u_2 \end{bmatrix} \\ &\implies r_3 = 1 \\ &\ddot{y}(x) = \begin{bmatrix} \sigma(\dot{x}_2 - \dot{x}_1) \\ (\dots) \end{bmatrix} = \begin{bmatrix} \sigma(\dots + u1 - \dot{x}_1) \\ (\dots) \end{bmatrix} \\ &\implies r_1 = 2\end{aligned}$$

The relative degree vector for this system and the selected output is then $r = [r_1, r_3] = [2, 1]$ and it is clear $\sum r_i = 3$. Using these r_i values, the following matrix and vector were constructed. Feedback linearizability is doubly confirmed by the global non-singularity of the $A(x)$ matrix.

$$A(x) = \begin{bmatrix} L_{f_1}L_{f_0}^{r_1-1}h_1 & L_{f_2}L_{f_0}^{r_1-1}h_1 \\ L_{f_1}L_{f_0}^{r_3-1}h_3 & L_{f_2}L_{f_0}^{r_3-1}h_3 \end{bmatrix} = \begin{bmatrix} \sigma & 0 \\ 0 & 1 \end{bmatrix}$$

$$b(x) = \begin{bmatrix} L_{f_0}^{r_1}h_1 \\ L_{f_0}^{r_3}h_3 \end{bmatrix} = \begin{bmatrix} \sigma^2(x_1 - x_2) + \sigma(x_1(\rho - x_3) - x_2) \\ x_1x_2 - \beta x_3 \end{bmatrix}$$

Using these two components, the essential variables $\alpha(x)$ and $\beta(x)$ were constructed using the following formulas:

$$\begin{aligned}\alpha(x) &= -A(x)^{-1}b(x) \\ \beta(x) &= A(x)^{-1}\end{aligned}$$

These were then combined to create the feedback stabilizing control input $u(x) = \alpha(x) + \beta(x)v$. The only missing component at this step is the appropriate v. This control variable was found after transforming the system into the feedback linearized z coordinate frame and calculating the LQR gain matrix. The appropriate coordinate transformation from x to z was determined through the construction of the $\Phi(x)$ set.

$$\Phi(x) = \{h_1, L^1_{f_0}h_1, h_3\} = \{x_1, -\sigma(x_1 - x_2), x_3\}$$
$$\therefore z_1 = x_1, z_2 = -\sigma(x_1 - x_2), z_3 = x_3$$

From here the Az and B feedback linearized matrices were formed.

$$Az = \left[\frac{\partial\Phi}{\partial x}(f_0(x) + [f_1(x), f_2(x)]\alpha(x))\right]$$
$$B = \left[\frac{\partial\Phi}{\partial x}[f_1(x), f_2(x)]\beta(x))\right]$$

where

$$\frac{\partial\Phi}{\partial x} = \begin{bmatrix} 1 & 0 & 0 \\ -\sigma & \sigma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The Az amd B matrices were then calculated to be the following:

$$Az = \begin{bmatrix} -\sigma(x_1 - x_2) \\ 0 \\ 0 \end{bmatrix} \implies A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$
$$B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

therefore the Feedback Linearized system becomes:

$$\dot{z} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} z + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} v$$

3.4. Trajectory Tracking

A Linear Quadratic Regulator was implemented on the Feedback Linearized system determined in the previous section in order to control the Lorenz system to track a desired trajectory. The following (standard) quadratic cost function was used.

$$J(x) = \int_0^\infty (x^T Q x + u^T R u)dt$$

with

$$Q = \begin{bmatrix} \delta & 0 & 0 \\ 0 & \delta & 0 \\ 0 & 0 & \delta \end{bmatrix} \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

where $\delta \in \mathbb{R}$ is a scaling factor in case the state requires an increase in weight. Assuming full-state feedback, $v = -Kz$, the LQR gain matrix K_{lqr} was calculated using the above A, B, Q, and R matrices in MATLAB using the *lqr()* function. For the matrix below a δ value of 10 was used.

$$K_{lqr} = \begin{bmatrix} 3.1623 & 4.0404 & 0 \\ 0 & 0 & 3.1623 \end{bmatrix}$$

Using $v = -Kz$, the linearized differential equation can be turned into the following closed-loop system.

$$\dot{z} = (A - BK)z$$

The trajectory tracking problem is posed in the x coordinate frame, which means the LQR control v needs to be transformed back into the nonlinear control variable u in order to command the system. Suppose there is a desired reference value $x_r \in \mathbb{R}^n$. Then to apply LQR control to stabilize the system at x_r, it would need to be transformed into z_r. Using the coordinate transformation matrix Φ, we can see $z_r = \Phi x_r$. Therefore, stabilizing z at this reference value changes the differential equation and informs what the appropriate v and subsequent u should be.

$$\dot{z} = (A - BK)z \longrightarrow \dot{z} = (A - BK)(z - z_r)$$
$$\implies v = -K(z - z_r)$$
$$\implies u = \alpha(x) - \beta(x)K(z - z_r)$$

The u determined above was utilized in conjunction with f_0, f_1 and f_2 in the control affine form of the nonlinear differential equation to implement control. A circular orbit in the x-y plane at an affine shift of $z = 3$ was the desired trajectory. The system was initialized at the point (2,1,0) and commanded to track this circular trajectory. As seen in Figures 2-4, the trajectory tracking effort was moderately successful. The orbit is confined to the x-y plane, but there is a notable eccentricity to the orbit relative to the desired path in red. This eccentric orbit does, however, appear to be stable for the implemented time horizon (T = 30π).

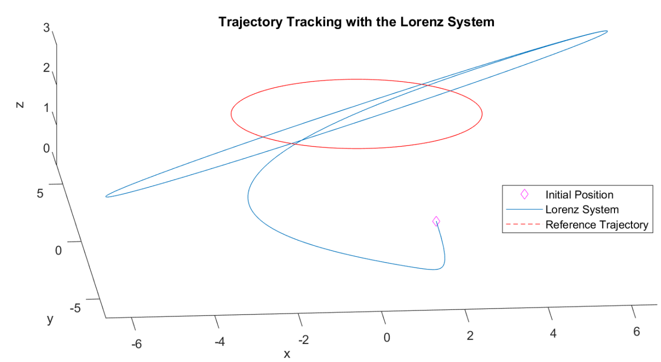


Figure 2. View 1: Controlled Lorenz System Attempting to track a circular orbit (3cos(t),3sin(t),3)

4. Discussion and Conclusions

Overall, the results of this study are reasonable and have great promise for future improvements. The controllability and observability of the system was definitive and required the minimal number of Lie Algebra elements possible to show it. That is, extensive sets of Lie brackets were not required to show $span\{\cdot\} = \mathbb{R}^3$ for $\mathcal{I}$ and $d\mathcal{O}$. This means there were fewer opportunities for error and therefore a higher level of confidence in the accuracy of the results. Feedback Linearization required a decent amount of effort,

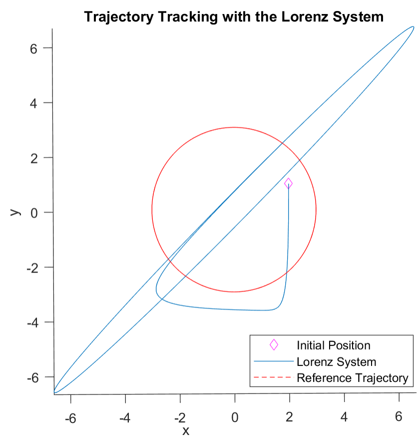


Figure 3. View 2: Controlled Lorenz System Attempting to track a circular orbit (3cos(t),3sin(t),3)

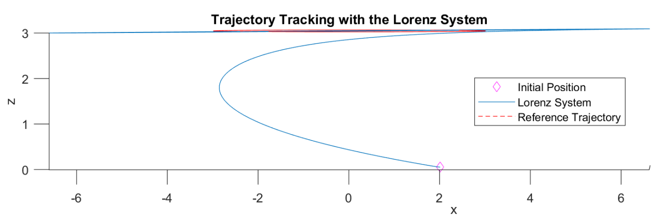


Figure 4. View 3: Controlled Lorenz System Attempting to track a circular orbit (3cos(t),3sin(t),3)

but the overall process was relatively straightforward for a MIMO system given the nature of the Lorenz system. The resulting A and B matrices were clearly controllable so it was reasonable to assume the LQR controller obtained from them would optimally stabilize the linearized system. The real complications occurred in the trajectory tracking section, as evident in Figures 2-4. The system did not exactly track the desired trajectory, but it did come close. The state trajectory was contained in the appropriate plane and had an eccentric version of the desired orbit. This eccentricity could be caused by a number of factors. One primary cause could be an erroneous implementation of reference point tracking with LQR. Perhaps substituting $z - z_r$ for z in the linearized differential equation is not adequate for a time or state varying z_r. Another possibility is that the control vector v was not properly translated back into u. An additional source of error is the method used to numerically integrate the system. The non-linear control affine system was numerically integrated using Forward Euler. A more robust approach would be to implement *ode45()* to numerically integrate the system. Fewer errors would arise and propagate using this function as opposed to Forward Euler. Each of these points, and likely more, would have to be investigated to determine the right corrective action to ensure the Lorenz system accurately tracks the desired trajectory.

Acknowledgement

I would like to thank Natalie Brace for her instrumental help in connecting the dots on how to implement feedback linearization on a MIMO system and the basic steps for utilizing the control in a meaningful way. She had to endure a great many questions.

References

[1] En.wikipedia.org. (2018). *Lorenz system*. [online] Available at: https://en.wikipedia.org/wiki/Lorenz_system [Accessed 24 May 2018].

AA 580 - Project - Nathan Isaman

```
clear all; close all; clc

syms r s b x1 x2 x3

states = [x1;x2;x3];

f0 = [s*(x2-x1);
      x1*(r-x3) - x2;
      x1*x2 - b*x3];
f1 = [0;1;0];
f2 = [0;0;1];

df0 = jacobian(f0,states);
df1 = jacobian(f1,states);
df2 = jacobian(f2,states);

F = horzcat(f1,f2);

% [f0,f1]
f0f1 = df1*f0 - df0*f1
% [f0,f2]
f0f2 = df2*f0 - df0*f2

F = horzcat(F,f0f1,f0f2)
Ftest1 = subs(F,transpose(states),{0,0,0})

rank(Ftest1)

df0f1 = jacobian(f0f1,states)
df0f2 = jacobian(f0f2,states)
% [f0,[f0,f1]]
f0f0f1 = df0f1*f0 - df0*f0f1
% [f0,[f0,f2]]
f0f0f2 = df0f2*f0 - df0*f0f2

% Little Observability Algebras (Non-Trivial Cases)
h1 = x1;
h2 = x2;
h3 = x3;

% Testing with just h1
Lf0h1 = jacobian(h1,states)*f0
LLf0h1 = jacobian(Lf0h1,states)*f0
LLLf0h1 = jacobian(LLf0h1,states)*f0
LLLLf0h1 = jacobian(LLLf0h1,states)*f0
O1 = horzcat(h1,Lf0h1,LLf0h1,LLLf0h1,LLLLf0h1)
dO1 = jacobian(O1,states)
subs(dO1,transpose(states),{0,0,0})
rank(subs(dO1,transpose(states),{0,0,0}))
```

```
% Testing with just h2
Lf0h2 = jacobian(h2,states)*f0
LLf0h2 = jacobian(Lf0h2,states)*f0
LLLf0h2 = jacobian(LLf0h2,states)*f0
LLLLf0h2 = jacobian(LLLf0h2,states)*f0
O2 = horzcat(h2,Lf0h2,LLf0h2,LLLf0h2,LLLLf0h2)
dO2 = jacobian(O2,states)
subs(dO2,transpose(states),{0,0,0})
rank(subs(dO2,transpose(states),{0,0,0}))

% Testing with just h3
Lf0h3 = jacobian(h3,states)*f0
LLf0h3 = jacobian(Lf0h3,states)*f0
LLLf0h3 = jacobian(LLf0h3,states)*f0
LLLLf0h3 = jacobian(LLLf0h3,states)*f0
O3 = horzcat(h3,Lf0h3,LLf0h3,LLLf0h3,LLLLf0h3)
dO3 = jacobian(O3,states)
subs(dO3,transpose(states),{0,0,0})
rank(subs(dO3,transpose(states),{0,0,0}))

% Testing with h1 and h2
fprintf('dO12 with h1 and h2: \n')
O12 = horzcat(O1,O2);
dO12 = jacobian(O12,states)
subs(dO12,transpose(states),{0,0,0})
rank(subs(dO12,transpose(states),{0,0,0}))

% Testing with h1 and h3
fprintf('dO13 with h1 and h3: \n')
O13 = horzcat(O1,O3);
dO13 = jacobian(O13,states)
subs(dO13,transpose(states),{0,0,0})
rank(subs(dO13,transpose(states),{0,0,0}))

% Testing with h2 and h3
fprintf('dO23 with h2 and h3: \n')
O23 = horzcat(O2,O3);
dO23 = jacobian(O23,states)
subs(dO23,transpose(states),{0,0,0})
rank(subs(dO23,transpose(states),{0,0,0}))

% Big Observability Algebra tests for nontrivial observation cases
% Testing with just h1
fprintf('Big Obs Alg with just h1\n')
Lf1h1 = jacobian(h1,states)*f1
LLf1h1 = jacobian(Lf1h1,states)*f1
Lf2h1 = jacobian(h1,states)*f2
LLf2h1 = jacobian(Lf2h1,states)*f2
O1B =
 horzcat(h1,Lf0h1,LLf0h1,LLLf0h1,LLLf0h1,Lf1h1,LLf1h1,Lf2h1,LLf2h1)
dO1B = jacobian(O1B,states)
subs(dO1B,transpose(states),{0,0,0})
rank(subs(dO1B,transpose(states),{0,0,0}))
```

```
% Testing with just h2
fprintf('Big Obs Alg with just h2\n')
Lf1h2 = jacobian(h2,states)*f1
LLf1h2 = jacobian(Lf1h2,states)*f1
Lf2h2 = jacobian(h2,states)*f2
LLf2h2 = jacobian(Lf2h2,states)*f2
O2B =
 horzcat(h2,Lf0h2,LLf0h2,LLLf0h2,LLLf0h2,Lf1h2,LLf1h2,Lf2h2,LLf2h2)
dO2B = jacobian(O2B,states)
subs(dO2B,transpose(states),{0,0,0})
rank(subs(dO2B,transpose(states),{0,0,0}))

% Testing with just h3
fprintf('Big Obs Alg with just h3\n')
Lf1h3 = jacobian(h3,states)*f1
LLf1h3 = jacobian(Lf1h3,states)*f1
Lf2h3 = jacobian(h3,states)*f2
LLf2h3 = jacobian(Lf2h3,states)*f2
O3B =
 horzcat(h3,Lf0h3,LLf0h3,LLLf0h3,LLLf0h3,Lf1h3,LLf1h3,Lf2h3,LLf2h3)
dO3B = jacobian(O3B,states)
subs(dO3B,transpose(states),{0,0,0})
rank(subs(dO3B,transpose(states),{0,0,0}))

% Testing with h1 and h2
fprintf('Big Obs Alg with h1 and h2\n')
O12B = horzcat(O1B,O2B)
dO12B = jacobian(O12B,states)
subs(dO12B,transpose(states),{0,0,0})
rank(subs(dO12B,transpose(states),{0,0,0}))

% Testing with h1 and h3
fprintf('Big Obs Alg with h1 and h3\n')
O13B = horzcat(O1B,O3B)
dO13B = jacobian(O13B,states)
subs(dO13B,transpose(states),{0,0,0})
rank(subs(dO13B,transpose(states),{0,0,0}))

% Testing with h2 and h3
fprintf('Big Obs Alg with h2 h3\n')
O23B = horzcat(O2B,O3B)
dO23B = jacobian(O23B,states)
subs(dO23B,transpose(states),{0,0,0})
rank(subs(dO23B,transpose(states),{0,0,0}))

% Testing for Feedback Linearizability
fprintf('\nFeedback Linearization: ')
fprintf('\nDistributions Delta_0, Delta_1, and Delta_2')
D0 = horzcat(f1,f2)
D1 = horzcat(D0,f0f1,f0f2)
D2 = horzcat(D1,f0f0f1,f0f0f2)

fprintf('\nDistributions in the neighborhood of [0 0 0]')
fprintf('\nDelta_0: ')
```

```
subs(D0,transpose(states),{0,0,0})
fprintf('\nDelta_1: ')
subs(D1,transpose(states),{0,0,0})
fprintf('\nDelta_2: ')
subs(D2,transpose(states),{0,0,0})

fprintf('Ranks of Delta_0, Delta_1, Delta_2')
rank(subs(D0,transpose(states),{0,0,0}))
rank(subs(D1,transpose(states),{0,0,0}))
rank(subs(D2,transpose(states),{0,0,0}))

% Testing the involutivity of Delta_0 and Delta_1
fprintf('Testing the involutivity of Delta_0 :\n')
f1f2 = df2*f1 - df1*f2
fprintf('Testing the involutivity of Delta_1 :\n')
f1f0f1 = df0f1*f1 - df1*f0f1
f1f0f2 = df0f2*f1 - df1*f0f2
f2f0f1 = df0f1*f2 - df2*f0f1
f2f0f2 = df0f2*f2 - df2*f0f2

% Calculating Vector Relative Degree for Feedback Linearization
%h1
L1f0h1 = jacobian(h1,states)*f0;
L2f0h1 = jacobian(L1f0h1,states)*f0;
%h2
L1f0h2 = jacobian(h2,states)*f0;
L2f0h2 = jacobian(L1f0h2,states)*f0;
%h3
L1f0h3 = jacobian(h3,states)*f0;
L2f0h3 = jacobian(L1f0h3,states)*f0;
fprintf('\nTesting 1s order with f1')
Lf1L1f0h1 = jacobian(L1f0h1,states)*f1
Lf1L1f0h2 = jacobian(L1f0h2,states)*f1
Lf1L1f0h3 = jacobian(L1f0h3,states)*f1
fprintf('\nTesting 1s order with f2')
Lf2L1f0h1 = jacobian(L1f0h1,states)*f2
Lf2L1f0h2 = jacobian(L1f0h2,states)*f2
Lf2L1f0h3 = jacobian(L1f0h3,states)*f2

fprintf('\nTesting 2nd order with f1')
Lf1L2f0h1 = jacobian(L2f0h1,states)*f1
Lf1L2f0h2 = jacobian(L2f0h2,states)*f1
Lf1L2f0h3 = jacobian(L2f0h3,states)*f1
fprintf('\nTesting 2nd order with f2')
Lf2L2f0h1 = jacobian(L2f0h1,states)*f2
Lf2L2f0h2 = jacobian(L2f0h2,states)*f2
Lf2L2f0h3 = jacobian(L2f0h3,states)*f2

Lf1L0f0h1 = jacobian(h1,states)*f1;
Lf2L0f0h1 = jacobian(h1,states)*f2;
Lf1L0f0h3 = jacobian(h3,states)*f1;
Lf2L0f0h3 = jacobian(h3,states)*f2;
```

```
f0f1 =

  -s
   1
 -x1


f0f2 =

  0
 x1
  b


F =

[ 0, 0,  -s,  0]
[ 1, 0,   1, x1]
[ 0, 1, -x1,  b]


Ftest1 =

[ 0, 0, -s, 0]
[ 1, 0,  1, 0]
[ 0, 1,  0, b]


ans =

     3


df0f1 =

[  0, 0, 0]
[  0, 0, 0]
[ -1, 0, 0]


df0f2 =

[ 0, 0, 0]
[ 1, 0, 0]
[ 0, 0, 0]


f0f0f1 =

                      - s^2 - s
        - x1^2 + s*(r - x3) + 1
 s*x2 - b*x1 - x1 + s*(x1 - x2)
```

```
f0f0f2 =

                   -s*x1
 x1 + b*x1 - s*(x1 - x2)
              b^2 - x1^2


Lf0h1 =

-s*(x1 - x2)


LLf0h1 =

(x1 - x2)*s^2 + (x1*(r - x3) - x2)*s


LLLf0h1 =

(x2 - x1*(r - x3))*(s^2 + s) - s*(s^2 + (r - x3)*s)*(x1 - x2) +
 s*x1*(b*x3 - x1*x2)


LLLLf0h1 =

s*(x1 - x2)*(s*(s^2 + (r - x3)*s) + (s^2 + s)*(r - x3) - s*(b*x3 -
 x1*x2) + s*x1*x2) - (b*x3 - x1*x2)*(x1*(s^2 + s) + s^2*(x1 - x2) +
 b*s*x1) - (x2 - x1*(r - x3))*(s + s*(s^2 + (r - x3)*s) - s*x1^2 +
 s^2)


O1 =

[ x1, -s*(x1 - x2), (x1 - x2)*s^2 + (x1*(r - x3) - x2)*s, (x2 - x1*(r
 - x3))*(s^2 + s) - s*(s^2 + (r - x3)*s)*(x1 - x2) + s*x1*(b*x3 -
 x1*x2), s*(x1 - x2)*(s*(s^2 + (r - x3)*s) + (s^2 + s)*(r - x3) -
 s*(b*x3 - x1*x2) + s*x1*x2) - (b*x3 - x1*x2)*(x1*(s^2 + s) + s^2*(x1
 - x2) + b*s*x1) - (x2 - x1*(r - x3))*(s + s*(s^2 + (r - x3)*s) -
 s*x1^2 + s^2)]


dO1 =

[

                                                 1,


                                            0,

                                                      0]
[
```

```
                                                   -s,


                                               s,

                                                          0]
[


                                        s^2 + (r - x3)*s,


                                          - s^2 - s,

                                                  -s*x1]
[

                                                         s*(b*x3 - x1*x2) -
 (s^2 + s)*(r - x3) - s*(s^2 + (r - x3)*s) - s*x1*x2,


       s + s*(s^2 + (r - x3)*s) - s*x1^2 + s^2,

                x1*(s^2 + s) + s^2*(x1 - x2) + b*s*x1]
[ x2*(x1*(s^2 + s) + s^2*(x1 - x2) + b*s*x1) - (b*x3 - x1*x2)*(s +
 b*s + 2*s^2) + (r - x3)*(s + s*(s^2 + (r - x3)*s) - s*x1^2 + s^2)
 + s*(s*(s^2 + (r - x3)*s) + (s^2 + s)*(r - x3) - s*(b*x3 - x1*x2) +
 s*x1*x2) + 2*s^2*x2*(x1 - x2) + 2*s*x1*(x2 - x1*(r - x3)), s^2*(b*x3
 - x1*x2) - s + x1*(x1*(s^2 + s) + s^2*(x1 - x2) + b*s*x1) - s*(s^2 +
 (r - x3)*s) + s*x1^2 - s^2 - s*(s*(s^2 + (r - x3)*s) + (s^2 + s)*(r
 - x3) - s*(b*x3 - x1*x2) + s*x1*x2) + 2*s^2*x1*(x1 - x2), s^2*(x2 -
 x1*(r - x3)) - b*(x1*(s^2 + s) + s^2*(x1 - x2) + b*s*x1) - x1*(s +
 s*(s^2 + (r - x3)*s) - s*x1^2 + s^2) - s*(x1 - x2)*(s + b*s + 2*s^2)]


ans =

[                                                              1,
                                                       0, 0]
[                                                             -s,
                                                       s, 0]
[                                                       s^2 + r*s,
                                               - s^2 - s, 0]
[                                    - s*(s^2 + r*s) - r*(s^2 + s),
                                  s + s*(s^2 + r*s) + s^2, 0]
[ r*(s + s*(s^2 + r*s) + s^2) + s*(s*(s^2 + r*s) + r*(s^2 + s)), - s -
 s*(s^2 + r*s) - s^2 - s*(s*(s^2 + r*s) + r*(s^2 + s)), 0]


ans =

    2
```

```
Lf0h2 =

x1*(r - x3) - x2


LLf0h2 =

x2 - x1*(r - x3) + x1*(b*x3 - x1*x2) - s*(r - x3)*(x1 - x2)


LLLf0h2 =

s*(x1 - x2)*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) - (b*x3 -
 x1*x2)*(x1 + b*x1 + s*(x1 - x2)) - (x2 - x1*(r - x3))*(- x1^2 + s*(r
 - x3) + 1)


LLLLf0h2 =

(b*x3 - x1*x2)*(b*(x1 + b*x1 + s*(x1 - x2)) - s*(x2 - x1*(r - x3)) +
 x1*(- x1^2 + s*(r - x3) + 1) + s*(x1 - x2)*(b + s + 1)) - (x2 - x1*(r
 - x3))*(x1*(x1 + b*x1 + s*(x1 - x2)) - s*(r - x3) - s*(r - x3 - b*x3
 + 2*x1*x2 + s*(r - x3)) + s*(b*x3 - x1*x2) + x1^2 + 2*s*x1*(x1 - x2)
 - 1) - s*(x1 - x2)*((r - x3)*(- x1^2 + s*(r - x3) + 1) + 2*x1*(x2 -
 x1*(r - x3)) + x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3 - x1*x2)*(b + s +
 1) + s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + 2*s*x2*(x1 - x2))


O2 =

[ x2, x1*(r - x3) - x2, x2 - x1*(r - x3) + x1*(b*x3 - x1*x2) - s*(r
 - x3)*(x1 - x2), s*(x1 - x2)*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3))
 - (b*x3 - x1*x2)*(x1 + b*x1 + s*(x1 - x2)) - (x2 - x1*(r - x3))*(-
 x1^2 + s*(r - x3) + 1), (b*x3 - x1*x2)*(b*(x1 + b*x1 + s*(x1 - x2)) -
 s*(x2 - x1*(r - x3)) + x1*(- x1^2 + s*(r - x3) + 1) + s*(x1 - x2)*(b
 + s + 1)) - (x2 - x1*(r - x3))*(x1*(x1 + b*x1 + s*(x1 - x2)) - s*(r
 - x3) - s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + s*(b*x3 - x1*x2)
 + x1^2 + 2*s*x1*(x1 - x2) - 1) - s*(x1 - x2)*((r - x3)*(- x1^2 + s*(r
 - x3) + 1) + 2*x1*(x2 - x1*(r - x3)) + x2*(x1 + b*x1 + s*(x1 - x2)) -
 (b*x3 - x1*x2)*(b + s + 1) + s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3))
 + 2*s*x2*(x1 - x2))]


dO2 =

[
```

0,

1,

0]
[

r - x3,

-1,

-x1]
[

x3 - r + b*x3 - 2*x1*x2 - s*(r - x3),

- x1^2 + s(r - x3) + 1,*

*x1 + b*x1 + s*(x1 - x2)]*

[

(r - x3)(- x1^2 + s*(r - x3) + 1) + 2*x1*(x2 - x1*(r - x3)) + x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3 - x1*x2)*(b + s + 1) + s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + 2*s*x2*(x1 - x2),*

x1(x1 + b*x1 + s*(x1 - x2)) - s*(r - x3) - s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + s*(b*x3 - x1*x2) + x1^2 + 2*s*x1*(x1 - x2) - 1,*

s(x2 - x1*(r - x3)) - b*(x1 + b*x1 + s*(x1 - x2)) - x1*(- x1^2 + s*(r - x3) + 1) - s*(x1 - x2)*(b + s + 1)]*

*[(b*x3 - x1*x2)*(- 3*x1^2 + s*(b + s + 1) + 2*s*(r - x3) + b*(b + s + 1) + 1) - x2*(b*(x1 + b*x1 + s*(x1 - x2)) - s*(x2 - x1*(r - x3)) + x1*(- x1^2 + s*(r - x3) + 1) + s*(x1 - x2)*(b + s + 1)) - (x2 - x1*(r - x3))*(3*x1 + x1*(b + s + 1) + b*x1 + 2*s*x1 - 3*s*x2 + 3*s*(x1 - x2)) + (r - x3)*(x1*(x1 + b*x1 + s*(x1 - x2)) - s*(r - x3) - s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + s*(b*x3 - x1*x2) + x1^2 + 2*s*x1*(x1 - x2) - 1) - s*((r - x3)*(- x1^2 + s*(r - x3) + 1) + 2*x1*(x2 - x1*(r - x3)) + x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3 - x1*x2)*(b + s + 1) + s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + 2*s*x2*(x1 - x2)) - s*(x1 - x2)*(2*x2 + 2*x2*(b + s + 1) + 4*s*x2 - 6*x1*(r - x3)), s*(r - x3) - (b*x3 - x1*x2)*(s + s*(b + s + 1) + b*s) - x1*(x1 + b*x1 + s*(x1 - x2)) - x1*(b*(x1 + b*x1 + s*(x1 - x2)) - s*(x2 - x1*(r - x3)) + x1*(- x1^2 + s*(r - x3) + 1) + s*(x1 - x2)*(b + s + 1)) + s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) - s*(b*x3 - x1*x2) - x1^2 + s*((r - x3)*(- x1^2 + s*(r - x3) + 1) + 2*x1*(x2 - x1*(r - x3)) + x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3 - x1*x2)*(b + s + 1) + s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + 2*s*x2*(x1 - x2)) - s*(x1 - x2)*(3*x1 + x1*(b + s + 1) + b*x1 + 2*s*x1 - 3*s*x2 + 3*s*(x1 - x2)) + 6*s*x1*(x2 - x1*(r - x3)) - 2*s*x1*(x1 - x2) + 1, b*(b*(x1 + b*x1 + s*(x1 - x2)) - s*(x2 - x1*(r - x3)) + x1*(- x1^2 + s*(r - x3)*

```
 + 1) + s*(x1 - x2)*(b + s + 1)) - x1*(x1*(x1 + b*x1 + s*(x1 - x2))
 - s*(r - x3) - s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + s*(b*x3 -
 x1*x2) + x1^2 + 2*s*x1*(x1 - x2) - 1) - (x2 - x1*(r - x3))*(s + s*(b
 + s + 1) + b*s) - 2*s*x1*(b*x3 - x1*x2) + s*(x1 - x2)*(- 3*x1^2 +
 s*(b + s + 1) + 2*s*(r - x3) + b*(b + s + 1) + 1)]


ans =

[                                                              0,
                                              1, 0]
[                                                              r,
                                             -1, 0]
[                                                      - r - r*s,
                                        r*s + 1, 0]
[                                   r*(r*s + 1) + s*(r + r*s),
                           - r*s - s*(r + r*s) - 1, 0]
[ - r*(r*s + s*(r + r*s) + 1) - s*(r*(r*s + 1) + s*(r + r*s)), r*s +
 s*(r*(r*s + 1) + s*(r + r*s)) + s*(r + r*s) + 1, 0]


ans =

     2


Lf0h3 =

x1*x2 - b*x3


LLf0h3 =

b*(b*x3 - x1*x2) - x1*(x2 - x1*(r - x3)) - s*x2*(x1 - x2)


LLLf0h3 =

(x2 - x1*(r - x3))*(x1 + b*x1 - s*x2 + s*(x1 - x2)) - (b^2 -
 x1^2)*(b*x3 - x1*x2) + s*(x1 - x2)*(x2 + b*x2 + s*x2 - 2*x1*(r - x3))


LLLLf0h3 =

- (x2 - x1*(r - x3))*(x1 - 2*s*(x2 - x1*(r - x3)) + x1*(b^2 - x1^2)
 + b*x1 - s*x2 + s*(x1 - x2) - s*(x2 + b*x2 + s*x2 - 2*x1*(r -
 x3)) + s*(x1 - x2)*(b + s + 1)) - (b*x3 - x1*x2)*(x1*(x1 + b*x1 -
 s*x2 + s*(x1 - x2)) - b*(b^2 - x1^2) + 2*s*x1*(x1 - x2)) - s*(x1
 - x2)*(x2*(b^2 - x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) +
 2*x1*(b*x3 - x1*x2) - (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2
 - x1*(r - x3))*(b + s + 1) - s*(2*r - 2*x3)*(x1 - x2))


O3 =
```

```
[ x3, x1*x2 - b*x3, b*(b*x3 - x1*x2) - x1*(x2 - x1*(r - x3)) -
 s*x2*(x1 - x2), (x2 - x1*(r - x3))*(x1 + b*x1 - s*x2 + s*(x1 - x2))
 - (b^2 - x1^2)*(b*x3 - x1*x2) + s*(x1 - x2)*(x2 + b*x2 + s*x2 -
 2*x1*(r - x3)), - (x2 - x1*(r - x3))*(x1 - 2*s*(x2 - x1*(r - x3)) +
 x1*(b^2 - x1^2) + b*x1 - s*x2 + s*(x1 - x2) - s*(x2 + b*x2 + s*x2 -
 2*x1*(r - x3)) + s*(x1 - x2)*(b + s + 1)) - (b*x3 - x1*x2)*(x1*(x1
 + b*x1 - s*x2 + s*(x1 - x2)) - b*(b^2 - x1^2) + 2*s*x1*(x1 - x2)) -
 s*(x1 - x2)*(x2*(b^2 - x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) +
 2*x1*(b*x3 - x1*x2) - (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2
 - x1*(r - x3))*(b + s + 1) - s*(2*r - 2*x3)*(x1 - x2))]


dO3 =

[

                                       0,

                                           0,

                    1]
[

                                      x2,
```

```
x1,

-b]
[

2*x1*(r - x3) - b*x2 - s*x2 - x2,

s*x2 - b*x1 - x1 - s*(x1 - x2),

b^2 - x1^2]
[

x2*(b^2 - x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + 2*x1*(b*x3
- x1*x2) - (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2 - x1*(r -
x3))*(b + s + 1) - s*(2*r - 2*x3)*(x1 - x2),

x1 - 2*s*(x2 - x1*(r
- x3)) + x1*(b^2 - x1^2) + b*x1 - s*x2 + s*(x1 - x2) - s*(x2 + b*x2 +
s*x2 - 2*x1*(r - x3)) + s*(x1 - x2)*(b + s + 1),

x1*(x1 + b*x1 - s*x2 + s*(x1 - x2)) - b*(b^2
- x1^2) + 2*s*x1*(x1 - x2)]
```

```
[ (r - x3)*(x1 - 2*s*(x2 - x1*(r - x3)) + x1*(b^2 - x1^2) + b*x1 -
 s*x2 + s*(x1 - x2) - s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + s*(x1 -
 x2)*(b + s + 1)) - (x2 - x1*(r - x3))*(b + s + s*(b + s + 1) + 2*s*(r
 - x3) + b^2 - 3*x1^2 + s*(2*r - 2*x3) + 1) - (b*x3 - x1*x2)*(x1 +
 x1*(b + s + 1) + 3*b*x1 + 2*s*x1 - s*x2 + 3*s*(x1 - x2)) - s*(x2*(b^2
 - x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + 2*x1*(b*x3 - x1*x2)
 - (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2 - x1*(r - x3))*(b
 + s + 1) - s*(2*r - 2*x3)*(x1 - x2)) + x2*(x1*(x1 + b*x1 - s*x2 +
 s*(x1 - x2)) - b*(b^2 - x1^2) + 2*s*x1*(x1 - x2)) + s*(x1 - x2)*(2*(r
 - x3)*(b + s + 1) - 2*b*x3 + 6*x1*x2 + 2*s*(2*r - 2*x3)), s*(x2*(b^2
 - x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + 2*x1*(b*x3 - x1*x2)
 - (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2 - x1*(r - x3))*(b
 + s + 1) - s*(2*r - 2*x3)*(x1 - x2)) - x1 + 2*s*(x2 - x1*(r - x3))
 - x1*(b^2 - x1^2) - b*x1 + s*x2 - s*(x1 - x2) + s*(x2 + b*x2 + s*x2
 - 2*x1*(r - x3)) + (4*s + 2*s*(b + s + 1))*(x2 - x1*(r - x3)) +
 x1*(x1*(x1 + b*x1 - s*x2 + s*(x1 - x2)) - b*(b^2 - x1^2) + 2*s*x1*(x1
 - x2)) - s*(x1 - x2)*(b + s + s*(b + s + 1) + 2*s*(r - x3) + b^2 -
 3*x1^2 + s*(2*r - 2*x3) + 1) + 4*s*x1*(b*x3 - x1*x2) - s*(x1 - x2)*(b
 + s + 1), 4*s*x1*(x2 - x1*(r - x3)) - b*(x1*(x1 + b*x1 - s*x2 + s*(x1
 - x2)) - b*(b^2 - x1^2) + 2*s*x1*(x1 - x2)) - x1*(x1 - 2*s*(x2 -
 x1*(r - x3)) + x1*(b^2 - x1^2) + b*x1 - s*x2 + s*(x1 - x2) - s*(x2
 + b*x2 + s*x2 - 2*x1*(r - x3)) + s*(x1 - x2)*(b + s + 1)) - s*(x1 -
 x2)*(x1 + x1*(b + s + 1) + 3*b*x1 + 2*s*x1 - s*x2 + 3*s*(x1 - x2))]


ans =

[ 0, 0,    1]
[ 0, 0,   -b]
[ 0, 0,  b^2]
[ 0, 0, -b^3]
[ 0, 0,  b^4]


ans =

     1

dO12 with h1 and h2:

dO12 =

[



                                                             1,
```

```
0,
0]
[
-s,
s,
0]
[
s^2 + (r - x3)*s,
```

- s^2 - s,

-s*x1]

[

s*(b*x3 - x1*x2)
- (s^2 + s)*(r - x3) - s*(s^2 + (r - x3)*s) - s*x1*x2,

s + s*(s^2 + (r - x3)*s) -
s*x1^2 + s^2,

x1*(s^2 + s) + s^2*(x1 - x2) + b*s*x1]

[

x2*(x1*(s^2 + s) + s^2*(x1 - x2) + b*s*x1) - (b*x3 - x1*x2)*(s
+ b*s + 2*s^2) + (r - x3)*(s + s*(s^2 + (r - x3)*s) - s*x1^2 + s^2)
+ s*(s*(s^2 + (r - x3)*s) + (s^2 + s)*(r - x3) - s*(b*x3 - x1*x2) +
s*x1*x2) + 2*s^2*x2*(x1 - x2) + 2*s*x1*(x2 - x1*(r - x3)),

s^2*(b*x3 - x1*x2) - s + x1*(x1*(s^2
+ s) + s^2*(x1 - x2) + b*s*x1) - s*(s^2 + (r - x3)*s) + s*x1^2 - s^2
- s*(s*(s^2 + (r - x3)*s) + (s^2 + s)*(r - x3) - s*(b*x3 - x1*x2) +
s*x1*x2) + 2*s^2*x1*(x1 - x2),

s^2*(x2 - x1*(r - x3)) - b*(x1*(s^2 + s) + s^2*(x1 - x2)

```
 + b*s*x1) - x1*(s + s*(s^2 + (r - x3)*s) - s*x1^2 + s^2) - s*(x1 -
 x2)*(s + b*s + 2*s^2)]
[

                                                         0,

      1,

                                                                 0]
[

                                                    r - x3,

     -1,

                                                              -x1]
[
```

*x3 - r + b*x3 - 2*x1*x2 - s*(r - x3),*

- x1^2 + s(r - x3) + 1,*

*x1 + b*x1 + s*(x1 - x2)]*
[

(r - x3)(- x1^2 + s*(r - x3) + 1) + 2*x1*(x2 - x1*(r - x3)) + x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3 - x1*x2)*(b + s + 1) + s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + 2*s*x2*(x1 - x2),*

x1(x1 + b*x1 + s*(x1 - x2)) - s*(r - x3) - s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + s*(b*x3 - x1*x2) + x1^2 + 2*s*x1*(x1 - x2) - 1,*

s(x2 - x1*(r - x3)) - b*(x1 + b*x1 + s*(x1 - x2)) - x1*(- x1^2 + s*(r - x3) + 1) - s*(x1 - x2)*(b + s + 1)]*
*[(b*x3 - x1*x2)*(- 3*x1^2 + s*(b + s + 1) + 2*s*(r - x3) + b*(b + s + 1) + 1) - x2*(b*(x1 + b*x1 + s*(x1 - x2)) - s*(x2 - x1*(r - x3)) + x1*(- x1^2 + s*(r - x3) + 1) + s*(x1 - x2)*(b + s + 1)) - (x2 - x1*(r - x3))*(3*x1 + x1*(b + s + 1) + b*x1 + 2*s*x1 - 3*s*x2 + 3*s*(x1 - x2)) + (r - x3)*(x1*(x1 + b*x1 + s*(x1 - x2)) - s*(r - x3) - s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + s*(b*x3 - x1*x2) + x1^2 + 2*s*x1*(x1 - x2) - 1) - s*((r - x3)*(- x1^2 + s*(r - x3) + 1) + 2*x1*(x2 - x1*(r - x3)) + x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3 - x1*x2)*(b + s + 1) + s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + 2*s*x2*(x1 - x2)) - s*(x1 - x2)*(2*x2 + 2*x2*(b + s + 1) + 4*s*x2 -*

```
6*x1*(r - x3)), s*(r - x3) - (b*x3 - x1*x2)*(s + s*(b + s + 1) + b*s)
- x1*(x1 + b*x1 + s*(x1 - x2)) - x1*(b*(x1 + b*x1 + s*(x1 - x2)) -
s*(x2 - x1*(r - x3)) + x1*(- x1^2 + s*(r - x3) + 1) + s*(x1 - x2)*(b
+ s + 1)) + s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) - s*(b*x3 -
x1*x2) - x1^2 + s*((r - x3)*(- x1^2 + s*(r - x3) + 1) + 2*x1*(x2 -
x1*(r - x3)) + x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3 - x1*x2)*(b + s
+ 1) + s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + 2*s*x2*(x1 - x2)) -
s*(x1 - x2)*(3*x1 + x1*(b + s + 1) + b*x1 + 2*s*x1 - 3*s*x2 + 3*s*(x1
- x2)) + 6*s*x1*(x2 - x1*(r - x3)) - 2*s*x1*(x1 - x2) + 1, b*(b*(x1 +
b*x1 + s*(x1 - x2)) - s*(x2 - x1*(r - x3)) + x1*(- x1^2 + s*(r - x3)
+ 1) + s*(x1 - x2)*(b + s + 1)) - x1*(x1*(x1 + b*x1 + s*(x1 - x2))
- s*(r - x3) - s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + s*(b*x3 -
x1*x2) + x1^2 + 2*s*x1*(x1 - x2) - 1) - (x2 - x1*(r - x3))*(s + s*(b
+ s + 1) + b*s) - 2*s*x1*(b*x3 - x1*x2) + s*(x1 - x2)*(- 3*x1^2 +
s*(b + s + 1) + 2*s*(r - x3) + b*(b + s + 1) + 1)]


ans =

[                                                           1,
                                                   0, 0]
[                                                          -s,
                                                   s, 0]
[                                                  s^2 + r*s,
                                         - s^2 - s, 0]
[                            - s*(s^2 + r*s) - r*(s^2 + s),
                               s + s*(s^2 + r*s) + s^2, 0]
[ r*(s + s*(s^2 + r*s) + s^2) + s*(s*(s^2 + r*s) + r*(s^2 + s)), - s -
 s*(s^2 + r*s) - s^2 - s*(s*(s^2 + r*s) + r*(s^2 + s)), 0]
[                                                           0,
                                                   1, 0]
[                                                           r,
                                                  -1, 0]
[                                                 - r - r*s,
                                            r*s + 1, 0]
[                                r*(r*s + 1) + s*(r + r*s),
                               - r*s - s*(r + r*s) - 1, 0]
[   - r*(r*s + s*(r + r*s) + 1) - s*(r*(r*s + 1) + s*(r + r*s)),
 r*s + s*(r*(r*s + 1) + s*(r + r*s)) + s*(r + r*s) + 1, 0]


ans =

     2

dO13 with h1 and h3:

dO13 =

[
```

1,

0,

0]
[

-s,

s,

0]
[

s^2 + (r - x3)*s,

- s^2 - s,

*-s*x1]*
[

s(b*x3 - x1*x2) - (s^2 +*
s)(r - x3) - s*(s^2 + (r - x3)*s) - s*x1*x2,*

s + s(s^2 + (r - x3)*s) - s*x1^2 + s^2,*

x1(s^2 +*
s) + s^2(x1 - x2) + b*s*x1]*
[

x2(x1*(s^2 + s) + s^2*(x1 - x2) + b*s*x1) - (b*x3 - x1*x2)*(s*
*+ b*s + 2*s^2) + (r - x3)*(s + s*(s^2 + (r - x3)*s) - s*x1^2 + s^2)*
+ s(s*(s^2 + (r - x3)*s) + (s^2 + s)*(r - x3) - s*(b*x3 - x1*x2) +*
*s*x1*x2) + 2*s^2*x2*(x1 - x2) + 2*s*x1*(x2 - x1*(r - x3)),*

s^2(b*x3 - x1*x2) - s + x1*(x1*(s^2 + s) + s^2*(x1 - x2) + b*s*x1) -*
s(s^2 + (r - x3)*s) + s*x1^2 - s^2 - s*(s*(s^2 + (r - x3)*s) + (s^2*
+ s)(r - x3) - s*(b*x3 - x1*x2) + s*x1*x2) + 2*s^2*x1*(x1 - x2),*

s^2(x2 - x1*(r - x3))*

```
 - b*(x1*(s^2 + s) + s^2*(x1 - x2) + b*s*x1) - x1*(s + s*(s^2 + (r -
 x3)*s) - s*x1^2 + s^2) - s*(x1 - x2)*(s + b*s + 2*s^2)]
[
                                        0,
                                            0,
                          1]
[
                                        x2,
                                            x1,
                         -b]
[
```

```
2*x1*(r - x3) - b*x2 - s*x2 - x2,

s*x2 - b*x1 - x1 - s*(x1 - x2),

b^2 - x1^2]
[

x2*(b^2 - x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + 2*x1*(b*x3
- x1*x2) - (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2 - x1*(r -
x3))*(b + s + 1) - s*(2*r - 2*x3)*(x1 - x2),

x1 - 2*s*(x2 - x1*(r
- x3)) + x1*(b^2 - x1^2) + b*x1 - s*x2 + s*(x1 - x2) - s*(x2 + b*x2 +
s*x2 - 2*x1*(r - x3)) + s*(x1 - x2)*(b + s + 1),

x1*(x1 + b*x1 - s*x2 + s*(x1 - x2)) - b*(b^2
- x1^2) + 2*s*x1*(x1 - x2)]
[ (r - x3)*(x1 - 2*s*(x2 - x1*(r - x3)) + x1*(b^2 - x1^2) + b*x1 -
s*x2 + s*(x1 - x2) - s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + s*(x1 -
x2)*(b + s + 1)) - (x2 - x1*(r - x3))*(b + s + s*(b + s + 1) + 2*s*(r
- x3) + b^2 - 3*x1^2 + s*(2*r - 2*x3) + 1) - (b*x3 - x1*x2)*(x1 +
x1*(b + s + 1) + 3*b*x1 + 2*s*x1 - s*x2 + 3*s*(x1 - x2)) - s*(x2*(b^2
- x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + 2*x1*(b*x3 - x1*x2)
- (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2 - x1*(r - x3))*(b
+ s + 1) - s*(2*r - 2*x3)*(x1 - x2)) + x2*(x1*(x1 + b*x1 - s*x2 +
s*(x1 - x2)) - b*(b^2 - x1^2) + 2*s*x1*(x1 - x2)) + s*(x1 - x2)*(2*(r
- x3)*(b + s + 1) - 2*b*x3 + 6*x1*x2 + 2*s*(2*r - 2*x3)), s*(x2*(b^2
- x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + 2*x1*(b*x3 - x1*x2)
- (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2 - x1*(r - x3))*(b
+ s + 1) - s*(2*r - 2*x3)*(x1 - x2)) - x1 + 2*s*(x2 - x1*(r - x3))
- x1*(b^2 - x1^2) - b*x1 + s*x2 - s*(x1 - x2) + s*(x2 + b*x2 + s*x2
```

```
- 2*x1*(r - x3)) + (4*s + 2*s*(b + s + 1))*(x2 - x1*(r - x3)) +
x1*(x1*(x1 + b*x1 - s*x2 + s*(x1 - x2)) - b*(b^2 - x1^2) + 2*s*x1*(x1
- x2)) - s*(x1 - x2)*(b + s + s*(b + s + 1) + 2*s*(r - x3) + b^2 -
3*x1^2 + s*(2*r - 2*x3) + 1) + 4*s*x1*(b*x3 - x1*x2) - s*(x1 - x2)*(b
+ s + 1), 4*s*x1*(x2 - x1*(r - x3)) - b*(x1*(x1 + b*x1 - s*x2 + s*(x1
- x2)) - b*(b^2 - x1^2) + 2*s*x1*(x1 - x2)) - x1*(x1 - 2*s*(x2 -
x1*(r - x3)) + x1*(b^2 - x1^2) + b*x1 - s*x2 + s*(x1 - x2) - s*(x2
+ b*x2 + s*x2 - 2*x1*(r - x3)) + s*(x1 - x2)*(b + s + 1)) - s*(x1 -
x2)*(x1 + x1*(b + s + 1) + 3*b*x1 + 2*s*x1 - s*x2 + 3*s*(x1 - x2))]


ans =

[                                                              1,
                                                      0,    0]
[                                                             -s,
                                                      s,    0]
[                                                      s^2 + r*s,
                                              - s^2 - s,    0]
[                                  - s*(s^2 + r*s) - r*(s^2 + s),
                               s + s*(s^2 + r*s) + s^2,    0]
[ r*(s + s*(s^2 + r*s) + s^2) + s*(s*(s^2 + r*s) + r*(s^2 + s)), - s -
 s*(s^2 + r*s) - s^2 - s*(s*(s^2 + r*s) + r*(s^2 + s)),    0]
[                                                              0,
                                                      0,    1]
[                                                              0,
                                                      0,   -b]
[                                                              0,
                                                      0,  b^2]
[                                                              0,
                                                      0, -b^3]
[                                                              0,
                                                      0,  b^4]


ans =

     3

dO23 with h2 and h3:

dO23 =

[



                                                  0,
```

1,

0]
[

r - x3,

-1,

-x1]
[

*x3 - r + b*x3 - 2*x1*x2 - s*(r - x3),*

- x1^2 + s*(r
- x3) + 1,

x1 + b*x1 + s*(x1 - x2)]
[

(r - x3)*(- x1^2 + s*(r - x3) + 1) + 2*x1*(x2 - x1*(r - x3)) +
x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3 - x1*x2)*(b + s + 1) + s*(r - x3
- b*x3 + 2*x1*x2 + s*(r - x3)) + 2*s*x2*(x1 - x2),

x1*(x1 + b*x1 + s*(x1 - x2)) - s*(r - x3) - s*(r - x3
- b*x3 + 2*x1*x2 + s*(r - x3)) + s*(b*x3 - x1*x2) + x1^2 + 2*s*x1*(x1
- x2) - 1,

s*(x2 - x1*(r - x3)) - b*(x1 + b*x1 + s*(x1 -
x2)) - x1*(- x1^2 + s*(r - x3) + 1) - s*(x1 - x2)*(b + s + 1)]
[(b*x3 - x1*x2)*(- 3*x1^2 + s*(b + s + 1) + 2*s*(r - x3) + b*(b
+ s + 1) + 1) - x2*(b*(x1 + b*x1 + s*(x1 - x2)) - s*(x2 - x1*(r -
x3)) + x1*(- x1^2 + s*(r - x3) + 1) + s*(x1 - x2)*(b + s + 1)) -
(x2 - x1*(r - x3))*(3*x1 + x1*(b + s + 1) + b*x1 + 2*s*x1 - 3*s*x2
+ 3*s*(x1 - x2)) + (r - x3)*(x1*(x1 + b*x1 + s*(x1 - x2)) - s*(r -
x3) - s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + s*(b*x3 - x1*x2)
+ x1^2 + 2*s*x1*(x1 - x2) - 1) - s*((r - x3)*(- x1^2 + s*(r - x3) +
1) + 2*x1*(x2 - x1*(r - x3)) + x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3
- x1*x2)*(b + s + 1) + s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) +
2*s*x2*(x1 - x2)) - s*(x1 - x2)*(2*x2 + 2*x2*(b + s + 1) + 4*s*x2 -
6*x1*(r - x3)), s*(r - x3) - (b*x3 - x1*x2)*(s + s*(b + s + 1) + b*s)
- x1*(x1 + b*x1 + s*(x1 - x2)) - x1*(b*(x1 + b*x1 + s*(x1 - x2)) -
s*(x2 - x1*(r - x3)) + x1*(- x1^2 + s*(r - x3) + 1) + s*(x1 - x2)*(b
+ s + 1)) + s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) - s*(b*x3 -
x1*x2) - x1^2 + s*((r - x3)*(- x1^2 + s*(r - x3) + 1) + 2*x1*(x2 -
x1*(r - x3)) + x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3 - x1*x2)*(b + s
+ 1) + s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + 2*s*x2*(x1 - x2)) -
s*(x1 - x2)*(3*x1 + x1*(b + s + 1) + b*x1 + 2*s*x1 - 3*s*x2 + 3*s*(x1
- x2)) + 6*s*x1*(x2 - x1*(r - x3)) - 2*s*x1*(x1 - x2) + 1, b*(b*(x1 +
b*x1 + s*(x1 - x2)) - s*(x2 - x1*(r - x3)) + x1*(- x1^2 + s*(r - x3)
+ 1) + s*(x1 - x2)*(b + s + 1)) - x1*(x1*(x1 + b*x1 + s*(x1 - x2))
- s*(r - x3) - s*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + s*(b*x3 -
x1*x2) + x1^2 + 2*s*x1*(x1 - x2) - 1) - (x2 - x1*(r - x3))*(s + s*(b

```
 + s + 1) + b*s) - 2*s*x1*(b*x3 - x1*x2) + s*(x1 - x2)*(- 3*x1^2 +
 s*(b + s + 1) + 2*s*(r - x3) + b*(b + s + 1) + 1)]
[
                                                          0,

                  0,

                                                                    1]
[
                                                         x2,

                 x1,

                                                                   -b]
[
```

*2*x1*(r - x3) - b*x2 - s*x2 - x2,*

*s*x2 - b*x1 - x1 - s*(x1 - x2),*

b^2 - x1^2]
[

x2(b^2 - x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + 2*x1*(b*x3 - x1*x2) - (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2 - x1*(r - x3))*(b + s + 1) - s*(2*r - 2*x3)*(x1 - x2),*

*x1 - 2*s*(x2 - x1*(r - x3)) + x1*(b^2 - x1^2) + b*x1 - s*x2 + s*(x1 - x2) - s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + s*(x1 - x2)*(b + s + 1),*

x1(x1 + b*x1 - s*x2 + s*(x1 - x2)) - b*(b^2 - x1^2) + 2*s*x1*(x1 - x2)]*
[(r - x3)(x1 - 2*s*(x2 - x1*(r - x3)) + x1*(b^2 - x1^2) + b*x1 - s*x2 + s*(x1 - x2) - s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + s*(x1 - x2)*(b + s + 1)) - (x2 - x1*(r - x3))*(b + s + s*(b + s + 1) + 2*s*(r - x3) + b^2 - 3*x1^2 + s*(2*r - 2*x3) + 1) - (b*x3 - x1*x2)*(x1 + x1*(b + s + 1) + 3*b*x1 + 2*s*x1 - s*x2 + 3*s*(x1 - x2)) - s*(x2*(b^2 - x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + 2*x1*(b*x3 - x1*x2) - (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2 - x1*(r - x3))*(b + s + 1) - s*(2*r - 2*x3)*(x1 - x2)) + x2*(x1*(x1 + b*x1 - s*x2 + s*(x1 - x2)) - b*(b^2 - x1^2) + 2*s*x1*(x1 - x2)) +*

```
s*(x1 - x2)*(2*(r - x3)*(b + s + 1) - 2*b*x3 + 6*x1*x2 + 2*s*(2*r -
2*x3)),                          s*(x2*(b^2 - x1^2) + s*(x2 + b*x2
+ s*x2 - 2*x1*(r - x3)) + 2*x1*(b*x3 - x1*x2) - (r - x3)*(x1 + b*x1
- s*x2 + s*(x1 - x2)) + (x2 - x1*(r - x3))*(b + s + 1) - s*(2*r -
2*x3)*(x1 - x2)) - x1 + 2*s*(x2 - x1*(r - x3)) - x1*(b^2 - x1^2) -
b*x1 + s*x2 - s*(x1 - x2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) +
(4*s + 2*s*(b + s + 1))*(x2 - x1*(r - x3)) + x1*(x1*(x1 + b*x1 - s*x2
+ s*(x1 - x2)) - b*(b^2 - x1^2) + 2*s*x1*(x1 - x2)) - s*(x1 - x2)*(b
+ s + s*(b + s + 1) + 2*s*(r - x3) + b^2 - 3*x1^2 + s*(2*r - 2*x3) +
1) + 4*s*x1*(b*x3 - x1*x2) - s*(x1 - x2)*(b + s + 1),
                                                        4*s*x1*(x2
- x1*(r - x3)) - b*(x1*(x1 + b*x1 - s*x2 + s*(x1 - x2)) - b*(b^2 -
x1^2) + 2*s*x1*(x1 - x2)) - x1*(x1 - 2*s*(x2 - x1*(r - x3)) + x1*(b^2
- x1^2) + b*x1 - s*x2 + s*(x1 - x2) - s*(x2 + b*x2 + s*x2 - 2*x1*(r -
x3)) + s*(x1 - x2)*(b + s + 1)) - s*(x1 - x2)*(x1 + x1*(b + s + 1) +
3*b*x1 + 2*s*x1 - s*x2 + 3*s*(x1 - x2))]


ans =

[                                                              0,
                                              1,     0]
[                                                              r,
                                             -1,     0]
[                                                      - r - r*s,
                                         r*s + 1,    0]
[                                   r*(r*s + 1) + s*(r + r*s),
                          - r*s - s*(r + r*s) - 1,   0]
[ - r*(r*s + s*(r + r*s) + 1) - s*(r*(r*s + 1) + s*(r + r*s)), r*s +
 s*(r*(r*s + 1) + s*(r + r*s)) + s*(r + r*s) + 1,    0]
[                                                              0,
                                              0,     1]
[                                                              0,
                                              0,    -b]
[                                                              0,
                                              0,   b^2]
[                                                              0,
                                              0,  -b^3]
[                                                              0,
                                              0,   b^4]


ans =

     3

Big Obs Alg with just h1

Lf1h1 =

0


LLf1h1 =
```

```
0


Lf2h1 =

0


LLf2h1 =

0


O1B =

[ x1, -s*(x1 - x2), (x1 - x2)*s^2 + (x1*(r - x3) - x2)*s, (x2 - x1*(r
 - x3))*(s^2 + s) - s*(s^2 + (r - x3)*s)*(x1 - x2) + s*x1*(b*x3 -
 x1*x2), (x2 - x1*(r - x3))*(s^2 + s) - s*(s^2 + (r - x3)*s)*(x1 - x2)
 + s*x1*(b*x3 - x1*x2), 0, 0, 0, 0]


dO1B =

[
  1,                                        0,
          0]
[
  -s,                                        s,
           0]
[                                                       s^2 + (r -
 x3)*s,                                - s^2 - s,
        -s*x1]
[ s*(b*x3 - x1*x2) - (s^2 + s)*(r - x3) - s*(s^2 + (r - x3)*s) -
 s*x1*x2, s + s*(s^2 + (r - x3)*s) - s*x1^2 + s^2, x1*(s^2 + s) +
 s^2*(x1 - x2) + b*s*x1]
[ s*(b*x3 - x1*x2) - (s^2 + s)*(r - x3) - s*(s^2 + (r - x3)*s) -
 s*x1*x2, s + s*(s^2 + (r - x3)*s) - s*x1^2 + s^2, x1*(s^2 + s) +
 s^2*(x1 - x2) + b*s*x1]
[
  0,                                        0,
          0]
[
  0,                                        0,
          0]
[
  0,                                        0,
          0]
[
  0,                                        0,
          0]


ans =
```

```
[                               1,                        0, 0]
[                              -s,                        s, 0]
[                      s^2 + r*s,               - s^2 - s, 0]
[ - s*(s^2 + r*s) - r*(s^2 + s), s + s*(s^2 + r*s) + s^2, 0]
[ - s*(s^2 + r*s) - r*(s^2 + s), s + s*(s^2 + r*s) + s^2, 0]
[                               0,                        0, 0]
[                               0,                        0, 0]
[                               0,                        0, 0]
[                               0,                        0, 0]


ans =

     2

Big Obs Alg with just h2

Lf1h2 =

1


LLf1h2 =

0


Lf2h2 =

0


LLf2h2 =

0


O2B =

[ x2, x1*(r - x3) - x2, x2 - x1*(r - x3) + x1*(b*x3 - x1*x2) - s*(r -
 x3)*(x1 - x2), s*(x1 - x2)*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3)) -
 (b*x3 - x1*x2)*(x1 + b*x1 + s*(x1 - x2)) - (x2 - x1*(r - x3))*(- x1^2
 + s*(r - x3) + 1), s*(x1 - x2)*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3))
 - (b*x3 - x1*x2)*(x1 + b*x1 + s*(x1 - x2)) - (x2 - x1*(r - x3))*(-
 x1^2 + s*(r - x3) + 1), 1, 0, 0, 0]


dO2B =

[

                                           0,
```

```
                                                   1,

       0]
[
                                                   r - x3,

                                                  -1,

     -x1]
[
          x3 - r + b*x3 - 2*x1*x2 - s*(r - x3),

                    - x1^2 + s*(r - x3) + 1,
                                                                      x1 + b*x1 +
 s*(x1 - x2)]
[ (r - x3)*(- x1^2 + s*(r - x3) + 1) + 2*x1*(x2 - x1*(r - x3)) +
 x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3 - x1*x2)*(b + s + 1) + s*(r -
 x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + 2*s*x2*(x1 - x2), x1*(x1 + b*x1
 + s*(x1 - x2)) - s*(r - x3) - s*(r - x3 - b*x3 + 2*x1*x2 + s*(r -
 x3)) + s*(b*x3 - x1*x2) + x1^2 + 2*s*x1*(x1 - x2) - 1, s*(x2 - x1*(r
 - x3)) - b*(x1 + b*x1 + s*(x1 - x2)) - x1*(- x1^2 + s*(r - x3) + 1) -
 s*(x1 - x2)*(b + s + 1)]
[ (r - x3)*(- x1^2 + s*(r - x3) + 1) + 2*x1*(x2 - x1*(r - x3)) +
 x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3 - x1*x2)*(b + s + 1) + s*(r -
 x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + 2*s*x2*(x1 - x2), x1*(x1 + b*x1
 + s*(x1 - x2)) - s*(r - x3) - s*(r - x3 - b*x3 + 2*x1*x2 + s*(r -
 x3)) + s*(b*x3 - x1*x2) + x1^2 + 2*s*x1*(x1 - x2) - 1, s*(x2 - x1*(r
 - x3)) - b*(x1 + b*x1 + s*(x1 - x2)) - x1*(- x1^2 + s*(r - x3) + 1) -
 s*(x1 - x2)*(b + s + 1)]
[

                                                        0,

                                                   0,

       0]
[

                                                        0,

                                                   0,

       0]
[

                                                        0,

                                                   0,

       0]
[
```

```
                                                 0,

                                            0,

       0]


ans =

[                           0,                          1, 0]
[                           r,                         -1, 0]
[                   - r - r*s,                    r*s + 1, 0]
[ r*(r*s + 1) + s*(r + r*s), - r*s - s*(r + r*s) - 1, 0]
[ r*(r*s + 1) + s*(r + r*s), - r*s - s*(r + r*s) - 1, 0]
[                           0,                          0, 0]
[                           0,                          0, 0]
[                           0,                          0, 0]
[                           0,                          0, 0]


ans =

     2

Big Obs Alg with just h3

Lf1h3 =

0


LLf1h3 =

0


Lf2h3 =

1


LLf2h3 =

0


O3B =

[ x3, x1*x2 - b*x3, b*(b*x3 - x1*x2) - x1*(x2 - x1*(r - x3)) -
 s*x2*(x1 - x2), (x2 - x1*(r - x3))*(x1 + b*x1 - s*x2 + s*(x1 - x2)) -
 (b^2 - x1^2)*(b*x3 - x1*x2) + s*(x1 - x2)*(x2 + b*x2 + s*x2 - 2*x1*(r
 - x3)), (x2 - x1*(r - x3))*(x1 + b*x1 - s*x2 + s*(x1 - x2)) - (b^2
 - x1^2)*(b*x3 - x1*x2) + s*(x1 - x2)*(x2 + b*x2 + s*x2 - 2*x1*(r -
 x3)), 0, 0, 1, 0]
```

```
dO3B =

[

                                          0,

                                          0,
                                             1]
[

                                         x2,

                                         x1,
                                            -b]
[

          2*x1*(r - x3) - b*x2 - s*x2 - x2,

            s*x2 - b*x1 - x1 - s*(x1 - x2),
                                b^2 - x1^2]
[ x2*(b^2 - x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + 2*x1*(b*x3
 - x1*x2) - (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2 - x1*(r -
 x3))*(b + s + 1) - s*(2*r - 2*x3)*(x1 - x2), x1 - 2*s*(x2 - x1*(r -
 x3)) + x1*(b^2 - x1^2) + b*x1 - s*x2 + s*(x1 - x2) - s*(x2 + b*x2 +
 s*x2 - 2*x1*(r - x3)) + s*(x1 - x2)*(b + s + 1), x1*(x1 + b*x1 - s*x2
 + s*(x1 - x2)) - b*(b^2 - x1^2) + 2*s*x1*(x1 - x2)]
[ x2*(b^2 - x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + 2*x1*(b*x3
 - x1*x2) - (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2 - x1*(r -
 x3))*(b + s + 1) - s*(2*r - 2*x3)*(x1 - x2), x1 - 2*s*(x2 - x1*(r -
 x3)) + x1*(b^2 - x1^2) + b*x1 - s*x2 + s*(x1 - x2) - s*(x2 + b*x2 +
 s*x2 - 2*x1*(r - x3)) + s*(x1 - x2)*(b + s + 1), x1*(x1 + b*x1 - s*x2
 + s*(x1 - x2)) - b*(b^2 - x1^2) + 2*s*x1*(x1 - x2)]
[

                                          0,

                                          0,
                                             0]
[

                                          0,

                                          0,
                                             0]
[

                                          0,

                                          0,
                                             0]
[
```

```
                                                      0,

                                                      0,
                                                         0]


ans =

[ 0, 0,    1]
[ 0, 0,   -b]
[ 0, 0,  b^2]
[ 0, 0, -b^3]
[ 0, 0, -b^3]
[ 0, 0,    0]
[ 0, 0,    0]
[ 0, 0,    0]
[ 0, 0,    0]


ans =

     1

Big Obs Alg with h1 and h2

O12B =

[ x1, -s*(x1 - x2), (x1 - x2)*s^2 + (x1*(r - x3) - x2)*s, (x2 - x1*(r
 - x3))*(s^2 + s) - s*(s^2 + (r - x3)*s)*(x1 - x2) + s*x1*(b*x3 -
 x1*x2), (x2 - x1*(r - x3))*(s^2 + s) - s*(s^2 + (r - x3)*s)*(x1 - x2)
 + s*x1*(b*x3 - x1*x2), 0, 0, 0, 0, x2, x1*(r - x3) - x2, x2 - x1*(r -
 x3) + x1*(b*x3 - x1*x2) - s*(r - x3)*(x1 - x2), s*(x1 - x2)*(r - x3
 - b*x3 + 2*x1*x2 + s*(r - x3)) - (b*x3 - x1*x2)*(x1 + b*x1 + s*(x1 -
 x2)) - (x2 - x1*(r - x3))*(- x1^2 + s*(r - x3) + 1), s*(x1 - x2)*(r -
 x3 - b*x3 + 2*x1*x2 + s*(r - x3)) - (b*x3 - x1*x2)*(x1 + b*x1 + s*(x1
 - x2)) - (x2 - x1*(r - x3))*(- x1^2 + s*(r - x3) + 1), 1, 0, 0, 0]


dO12B =

[

                                                        1,

                                                   0,

      0]
[

                                                       -s,

                                                   s,

      0]
```

```
[
                              s^2 + (r - x3)*s,
                                - s^2 - s,
   -s*x1]
[
                                                        s*(b*x3 - x1*x2) - (s^2 +
 s)*(r - x3) - s*(s^2 + (r - x3)*s) - s*x1*x2,
 s + s*(s^2 + (r - x3)*s) - s*x1^2 + s^2,
                                                     x1*(s^2 + s) + s^2*(x1 - x2)
 + b*s*x1]
[
                                                        s*(b*x3 - x1*x2) - (s^2 +
 s)*(r - x3) - s*(s^2 + (r - x3)*s) - s*x1*x2,
 s + s*(s^2 + (r - x3)*s) - s*x1^2 + s^2,
                                                     x1*(s^2 + s) + s^2*(x1 - x2)
 + b*s*x1]
[
                                                      0,
                                                 0,
       0]
[
                                                      0,
                                                 0,
       0]
[
                                                      0,
                                                 0,
       0]
[
                                                      0,
                                                 0,
       0]
[
                                                      0,
                                                 1,
```

```
        0]
[
                                                     r - x3,
                                                    -1,
      -x1]
[
          x3 - r + b*x3 - 2*x1*x2 - s*(r - x3),
                    - x1^2 + s*(r - x3) + 1,
                                                                x1 + b*x1 +
 s*(x1 - x2)]
[ (r - x3)*(- x1^2 + s*(r - x3) + 1) + 2*x1*(x2 - x1*(r - x3)) +
 x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3 - x1*x2)*(b + s + 1) + s*(r -
 x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + 2*s*x2*(x1 - x2), x1*(x1 + b*x1
 + s*(x1 - x2)) - s*(r - x3) - s*(r - x3 - b*x3 + 2*x1*x2 + s*(r -
 x3)) + s*(b*x3 - x1*x2) + x1^2 + 2*s*x1*(x1 - x2) - 1, s*(x2 - x1*(r
 - x3)) - b*(x1 + b*x1 + s*(x1 - x2)) - x1*(- x1^2 + s*(r - x3) + 1) -
 s*(x1 - x2)*(b + s + 1)]
[ (r - x3)*(- x1^2 + s*(r - x3) + 1) + 2*x1*(x2 - x1*(r - x3)) +
 x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3 - x1*x2)*(b + s + 1) + s*(r -
 x3 - b*x3 + 2*x1*x2 + s*(r - x3)) + 2*s*x2*(x1 - x2), x1*(x1 + b*x1
 + s*(x1 - x2)) - s*(r - x3) - s*(r - x3 - b*x3 + 2*x1*x2 + s*(r -
 x3)) + s*(b*x3 - x1*x2) + x1^2 + 2*s*x1*(x1 - x2) - 1, s*(x2 - x1*(r
 - x3)) - b*(x1 + b*x1 + s*(x1 - x2)) - x1*(- x1^2 + s*(r - x3) + 1) -
 s*(x1 - x2)*(b + s + 1)]
[
                                                          0,
                                                     0,
        0]
[
                                                          0,
                                                     0,
        0]
[
                                                          0,
                                                     0,
        0]
[
                                                          0,
```

```
                                             0,

       0]


ans =

[                              1,                        0, 0]
[                             -s,                        s, 0]
[                      s^2 + r*s,               - s^2 - s, 0]
[ - s*(s^2 + r*s) - r*(s^2 + s), s + s*(s^2 + r*s) + s^2, 0]
[ - s*(s^2 + r*s) - r*(s^2 + s), s + s*(s^2 + r*s) + s^2, 0]
[                              0,                        0, 0]
[                              0,                        0, 0]
[                              0,                        0, 0]
[                              0,                        0, 0]
[                              0,                        1, 0]
[                              r,                       -1, 0]
[                      - r - r*s,                  r*s + 1, 0]
[     r*(r*s + 1) + s*(r + r*s), - r*s - s*(r + r*s) - 1, 0]
[     r*(r*s + 1) + s*(r + r*s), - r*s - s*(r + r*s) - 1, 0]
[                              0,                        0, 0]
[                              0,                        0, 0]
[                              0,                        0, 0]
[                              0,                        0, 0]


ans =

     2

Big Obs Alg with h1 and h3

O13B =

[ x1, -s*(x1 - x2), (x1 - x2)*s^2 + (x1*(r - x3) - x2)*s, (x2 - x1*(r
 - x3))*(s^2 + s) - s*(s^2 + (r - x3)*s)*(x1 - x2) + s*x1*(b*x3 -
 x1*x2), (x2 - x1*(r - x3))*(s^2 + s) - s*(s^2 + (r - x3)*s)*(x1 - x2)
 + s*x1*(b*x3 - x1*x2), 0, 0, 0, 0, x3, x1*x2 - b*x3, b*(b*x3 - x1*x2)
 - x1*(x2 - x1*(r - x3)) - s*x2*(x1 - x2), (x2 - x1*(r - x3))*(x1 +
 b*x1 - s*x2 + s*(x1 - x2)) - (b^2 - x1^2)*(b*x3 - x1*x2) + s*(x1 -
 x2)*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)), (x2 - x1*(r - x3))*(x1 + b*x1
 - s*x2 + s*(x1 - x2)) - (b^2 - x1^2)*(b*x3 - x1*x2) + s*(x1 - x2)*(x2
 + b*x2 + s*x2 - 2*x1*(r - x3)), 0, 0, 1, 0]


dO13B =

[

                                                1,
```

```
                                        0,
                                          0]
[
                                       -s,
                                        s,
                                          0]
[
                             s^2 + (r - x3)*s,
                                   - s^2 - s,
                                       -s*x1]
[
                                          s*(b*x3 - x1*x2) - (s^2 +
 s)*(r - x3) - s*(s^2 + (r - x3)*s) - s*x1*x2,
    s + s*(s^2 + (r - x3)*s) - s*x1^2 + s^2,
       x1*(s^2 + s) + s^2*(x1 - x2) + b*s*x1]
[
                                          s*(b*x3 - x1*x2) - (s^2 +
 s)*(r - x3) - s*(s^2 + (r - x3)*s) - s*x1*x2,
    s + s*(s^2 + (r - x3)*s) - s*x1^2 + s^2,
       x1*(s^2 + s) + s^2*(x1 - x2) + b*s*x1]
[
                                        0,
                                        0,
                                          0]
[
                                        0,
                                        0,
                                          0]
[
                                        0,
                                        0,
                                          0]
[
                                        0,
                                        0,
                                          0]
[
                                        0,
```

```
                                                  0,
                                                     1]
[
                                                 x2,
                                                 x1,
                                                    -b]
[
          2*x1*(r - x3) - b*x2 - s*x2 - x2,
            s*x2 - b*x1 - x1 - s*(x1 - x2),
                                         b^2 - x1^2]
[ x2*(b^2 - x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + 2*x1*(b*x3
 - x1*x2) - (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2 - x1*(r -
 x3))*(b + s + 1) - s*(2*r - 2*x3)*(x1 - x2), x1 - 2*s*(x2 - x1*(r -
 x3)) + x1*(b^2 - x1^2) + b*x1 - s*x2 + s*(x1 - x2) - s*(x2 + b*x2 +
 s*x2 - 2*x1*(r - x3)) + s*(x1 - x2)*(b + s + 1), x1*(x1 + b*x1 - s*x2
 + s*(x1 - x2)) - b*(b^2 - x1^2) + 2*s*x1*(x1 - x2)]
[ x2*(b^2 - x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) + 2*x1*(b*x3
 - x1*x2) - (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2 - x1*(r -
 x3))*(b + s + 1) - s*(2*r - 2*x3)*(x1 - x2), x1 - 2*s*(x2 - x1*(r -
 x3)) + x1*(b^2 - x1^2) + b*x1 - s*x2 + s*(x1 - x2) - s*(x2 + b*x2 +
 s*x2 - 2*x1*(r - x3)) + s*(x1 - x2)*(b + s + 1), x1*(x1 + b*x1 - s*x2
 + s*(x1 - x2)) - b*(b^2 - x1^2) + 2*s*x1*(x1 - x2)]
[
                                                  0,
                                                  0,
                                                     0]
[
                                                  0,
                                                  0,
                                                     0]
[
                                                  0,
                                                  0,
                                                     0]
[
                                                  0,
                                                  0,
                                                     0]

ans =
```

```
[                             1,                       0,    0]
[                            -s,                       s,    0]
[                    s^2 + r*s,              - s^2 - s,    0]
[ - s*(s^2 + r*s) - r*(s^2 + s), s + s*(s^2 + r*s) + s^2,    0]
[ - s*(s^2 + r*s) - r*(s^2 + s), s + s*(s^2 + r*s) + s^2,    0]
[                             0,                       0,    0]
[                             0,                       0,    0]
[                             0,                       0,    0]
[                             0,                       0,    0]
[                             0,                       0,    1]
[                             0,                       0,   -b]
[                             0,                       0,  b^2]
[                             0,                       0, -b^3]
[                             0,                       0, -b^3]
[                             0,                       0,    0]
[                             0,                       0,    0]
[                             0,                       0,    0]
[                             0,                       0,    0]


ans =

     3

Big Obs Alg with h2 h3

O23B =

[ x2, x1*(r - x3) - x2, x2 - x1*(r - x3) + x1*(b*x3 - x1*x2) - s*(r
 - x3)*(x1 - x2), s*(x1 - x2)*(r - x3 - b*x3 + 2*x1*x2 + s*(r - x3))
 - (b*x3 - x1*x2)*(x1 + b*x1 + s*(x1 - x2)) - (x2 - x1*(r - x3))*(-
 x1^2 + s*(r - x3) + 1), s*(x1 - x2)*(r - x3 - b*x3 + 2*x1*x2 + s*(r
 - x3)) - (b*x3 - x1*x2)*(x1 + b*x1 + s*(x1 - x2)) - (x2 - x1*(r -
 x3))*(- x1^2 + s*(r - x3) + 1), 1, 0, 0, 0, x3, x1*x2 - b*x3, b*(b*x3
 - x1*x2) - x1*(x2 - x1*(r - x3)) - s*x2*(x1 - x2), (x2 - x1*(r -
 x3))*(x1 + b*x1 - s*x2 + s*(x1 - x2)) - (b^2 - x1^2)*(b*x3 - x1*x2)
 + s*(x1 - x2)*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)), (x2 - x1*(r -
 x3))*(x1 + b*x1 - s*x2 + s*(x1 - x2)) - (b^2 - x1^2)*(b*x3 - x1*x2) +
 s*(x1 - x2)*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)), 0, 0, 1, 0]


dO23B =

[

                                             0,

                                             1,

            0]
[

                                        r - x3,
```

```
                                        -1,

      -x1]
[

     x3 - r + b*x3 - 2*x1*x2 - s*(r - x3),

              - x1^2 + s*(r - x3) + 1,
                                                         x1 + b*x1
 + s*(x1 - x2)]
[ (r - x3)*(- x1^2 + s*(r - x3) + 1) + 2*x1*(x2 - x1*(r - x3)) +
 x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3 - x1*x2)*(b + s + 1) + s*(r - x3
 - b*x3 + 2*x1*x2 + s*(r - x3)) + 2*s*x2*(x1 - x2),      x1*(x1 + b*x1
 + s*(x1 - x2)) - s*(r - x3) - s*(r - x3 - b*x3 + 2*x1*x2 + s*(r -
 x3)) + s*(b*x3 - x1*x2) + x1^2 + 2*s*x1*(x1 - x2) - 1, s*(x2 - x1*(r
 - x3)) - b*(x1 + b*x1 + s*(x1 - x2)) - x1*(- x1^2 + s*(r - x3) + 1) -
 s*(x1 - x2)*(b + s + 1)]
[ (r - x3)*(- x1^2 + s*(r - x3) + 1) + 2*x1*(x2 - x1*(r - x3)) +
 x2*(x1 + b*x1 + s*(x1 - x2)) - (b*x3 - x1*x2)*(b + s + 1) + s*(r - x3
 - b*x3 + 2*x1*x2 + s*(r - x3)) + 2*s*x2*(x1 - x2),      x1*(x1 + b*x1
 + s*(x1 - x2)) - s*(r - x3) - s*(r - x3 - b*x3 + 2*x1*x2 + s*(r -
 x3)) + s*(b*x3 - x1*x2) + x1^2 + 2*s*x1*(x1 - x2) - 1, s*(x2 - x1*(r
 - x3)) - b*(x1 + b*x1 + s*(x1 - x2)) - x1*(- x1^2 + s*(r - x3) + 1) -
 s*(x1 - x2)*(b + s + 1)]
[

                                         0,

                                         0,

       0]
[

                                         0,

                                         0,

       0]
[

                                         0,

                                         0,

       0]
[

                                         0,

                                         0,

       0]
[
```

```
                                                            0,

                                                            0,

            1]
[

                                                           x2,

                                                           x1,

           -b]
[

            2*x1*(r - x3) - b*x2 - s*x2 - x2,

              s*x2 - b*x1 - x1 - s*(x1 - x2),

   b^2 - x1^2]
[    x2*(b^2 - x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) +
 2*x1*(b*x3 - x1*x2) - (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2
 - x1*(r - x3))*(b + s + 1) - s*(2*r - 2*x3)*(x1 - x2), x1 - 2*s*(x2 -
 x1*(r - x3)) + x1*(b^2 - x1^2) + b*x1 - s*x2 + s*(x1 - x2) - s*(x2 +
 b*x2 + s*x2 - 2*x1*(r - x3)) + s*(x1 - x2)*(b + s + 1),
                        x1*(x1 + b*x1 - s*x2 + s*(x1 - x2)) - b*(b^2 -
 x1^2) + 2*s*x1*(x1 - x2)]
[    x2*(b^2 - x1^2) + s*(x2 + b*x2 + s*x2 - 2*x1*(r - x3)) +
 2*x1*(b*x3 - x1*x2) - (r - x3)*(x1 + b*x1 - s*x2 + s*(x1 - x2)) + (x2
 - x1*(r - x3))*(b + s + 1) - s*(2*r - 2*x3)*(x1 - x2), x1 - 2*s*(x2 -
 x1*(r - x3)) + x1*(b^2 - x1^2) + b*x1 - s*x2 + s*(x1 - x2) - s*(x2 +
 b*x2 + s*x2 - 2*x1*(r - x3)) + s*(x1 - x2)*(b + s + 1),
                        x1*(x1 + b*x1 - s*x2 + s*(x1 - x2)) - b*(b^2 -
 x1^2) + 2*s*x1*(x1 - x2)]
[

                                                            0,

                                                            0,

            0]
[

                                                            0,

                                                            0,

            0]
[

                                                            0,

                                                            0,

            0]
```

```
[

                                                0,

                                                0,

            0]


ans =

[                         0,                         1,    0]
[                         r,                        -1,    0]
[                 - r - r*s,                   r*s + 1,    0]
[ r*(r*s + 1) + s*(r + r*s), - r*s - s*(r + r*s) - 1,    0]
[ r*(r*s + 1) + s*(r + r*s), - r*s - s*(r + r*s) - 1,    0]
[                         0,                         0,    0]
[                         0,                         0,    0]
[                         0,                         0,    0]
[                         0,                         0,    0]
[                         0,                         0,    1]
[                         0,                         0,   -b]
[                         0,                         0,  b^2]
[                         0,                         0, -b^3]
[                         0,                         0, -b^3]
[                         0,                         0,    0]
[                         0,                         0,    0]
[                         0,                         0,    0]
[                         0,                         0,    0]


ans =

     3


Feedback Linearization:
Distributions Delta_0, Delta_1, and Delta_2
D0 =

     0     0
     1     0
     0     1


D1 =

[ 0, 0,  -s,  0]
[ 1, 0,   1, x1]
[ 0, 1, -x1,  b]


D2 =
```

```
[ 0, 0,  -s,  0,                        - s^2 - s,                   -
s*x1]
[ 1, 0,   1, x1,         - x1^2 + s*(r - x3) + 1, x1 + b*x1 - s*(x1 -
 x2)]
[ 0, 1, -x1,  b, s*x2 - b*x1 - x1 + s*(x1 - x2),               b^2 -
 x1^2]


Distributions in the neighborhood of [0 0 0]
Delta_0:
ans =

[ 0, 0]
[ 1, 0]
[ 0, 1]


Delta_1:
ans =

[ 0, 0, -s, 0]
[ 1, 0,  1, 0]
[ 0, 1,  0, b]


Delta_2:
ans =

[ 0, 0, -s, 0, - s^2 - s,   0]
[ 1, 0,  1, 0,   r*s + 1,   0]
[ 0, 1,  0, b,         0, b^2]

Ranks of Delta_0, Delta_1, Delta_2
ans =

     2


ans =

     3


ans =

     3

Testing the involutivity of Delta_0 :

f1f2 =

 0
 0
 0
```

```
Testing the involutivity of Delta_1 :

f1f0f1 =

 0
 0
 0


f1f0f2 =

 0
 0
 0


f2f0f1 =

 0
 0
 0


f2f0f2 =

 0
 0
 0


Testing 1s order with f1
Lf1L1f0h1 =

s


Lf1L1f0h2 =

-1


Lf1L1f0h3 =

x1


Testing 1s order with f2
Lf2L1f0h1 =

0


Lf2L1f0h2 =
```

```
-x1


Lf2L1f0h3 =

-b


Testing 2nd order with f1
Lf1L2f0h1 =

- s^2 - s


Lf1L2f0h2 =

- x1^2 + s*(r - x3) + 1


Lf1L2f0h3 =

s*x2 - b*x1 - x1 - s*(x1 - x2)


Testing 2nd order with f2
Lf2L2f0h1 =

-s*x1


Lf2L2f0h2 =

x1 + b*x1 + s*(x1 - x2)


Lf2L2f0h3 =

b^2 - x1^2


Del = [Lf1L1f0h1,Lf2L1f0h1;Lf1L0f0h3,Lf2L0f0h3]
B = [L2f0h1;L1f0h3]

alpha = -inv(Del)*B
beta = inv(Del)

Phi = [h1;L1f0h1;h3]
dPhi = jacobian(Phi,states)
dphi = [1 0 0; -s s 0; 0 0 1];
Phi_ = [1 0 0; -s s 0; 0 0 1];
Phi_inv = inv(Phi_)

Az = simplify(dPhi*(f0 + [f1,f2]*alpha))
```

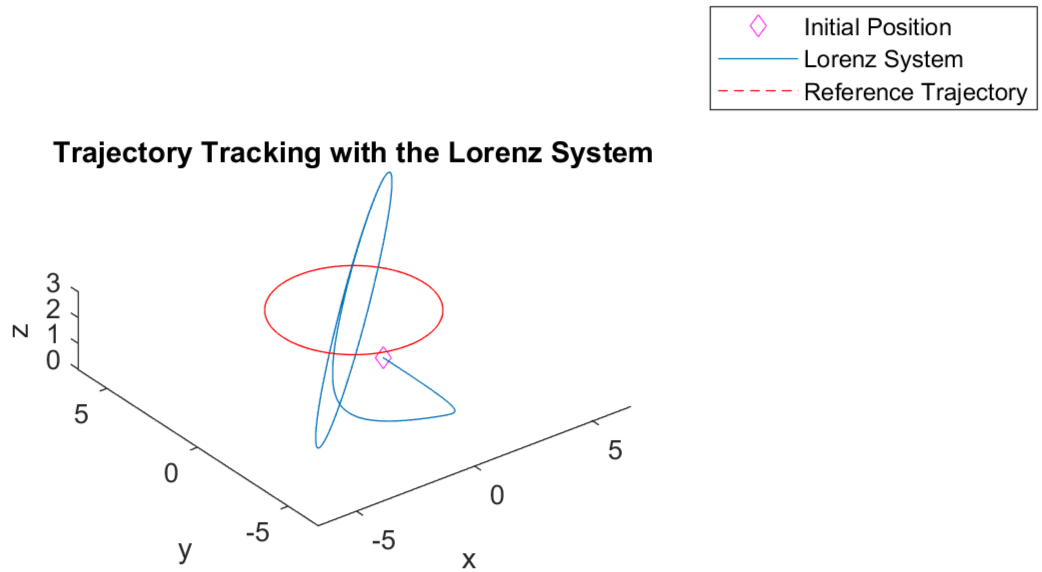
```
B = dPhi*[f1,f2]*beta

% LQR Design for Stabilization and Trajectory Tracking
Alqr = [0 1 0; 0 0 0; 0 0 0;]
Blqr = [0 0; 1 0; 0 1]
%Qlqr = eye(3)
Clqr = [1 0 0; 0 1 0; 0 0 1]        %Observing z1 and z3 (x1 and x3)
Qlqr = 10*transpose(Clqr)*Clqr
Rlqr = eye(2)

Klqr = lqr(Alqr,Blqr,Qlqr,Rlqr)

% Creating the state trajectory
% Setting sigma = 10, rho = 28 beta = 8/3
sigma = 10;
rho = 28;
beta = 8/3;

Phi_set = [1 0 0; -sigma sigma 0; 0 0 1];
Phi_set_inv = inv(Phi_set);
h = 0.001;
theta = 0:h:(30*pi);
x_init = [2;1;0];
x_ref_pt = [6;3;9];
z_ref_pt = Phi_set*x_ref_pt;
x_ref(:,1) = x_init;
for j = 1:length(theta)
   x_ref(:,j) = refFun(theta(j));
end
% Transforming the trajectory into z coords
for k = 1:length(theta)
    z_ref(:,k) = Phi_set*x_ref(:,k);
end

% z_state(:,1) = Phi_set*x_init;
% for i = 1:length(theta)
%     %z_state(:,i+1) = z_state(:,i) + h*((Alqr -
 Blqr*Klqr)*z_state(:,i) + Blqr*Klqr*z_ref(:,i));
%     z_state(:,i+1) = z_state(:,i) + h*(Alqr*z_state(:,i) -
 Blqr*Klqr*(z_state(:,i) - z_ref(:,i)));
%     %z_state(:,i+1) = z_state(:,i) + h*(Alqr*z_state(:,i) -
 Blqr*Klqr*(z_state(:,i) - z_ref_pt));
% end
% %Transforming the z state back into x state
% for j = 1:length(theta)
%    x_state(:,i) =  Phi_set_inv*z_state(:,i);
% end
%
% figure
% plot3(z_state(1,:),z_state(2,:),z_state(3,:))

x_state(:,1) = x_init;
for i = 1:length(theta)-1
   u = FBL_Ctrl(Phi_set,Phi_set_inv,Klqr,x_state(:,i),x_ref(:,i));
```

```
   x_state(:,i+1) = x_state(:,i) + h*(f0_fn(x_state(:,i)) + f1*u(1) +
 f2*u(2));
end
figure
plot3(x_init(1),x_init(2),x_init(3),'md'), hold on
plot3(x_state(1,:),x_state(2,:),x_state(3,:)), hold on
plot3(x_ref(1,:),x_ref(2,:),x_ref(3,:),'r--')
xlabel('x')
ylabel('y')
zlabel('z')
title('Trajectory Tracking with the Lorenz System')
legend('Initial Position','Lorenz System','Reference Trajectory')
axis equal
hold off


z_state(:,1) = Phi_set*x_init;
for k = 1:length(theta)
   z_state(:,k) = Phi_set*x_state(:,k);
end
% figure
% plot3(z_state(1,:),z_state(2,:),z_state(3,:))

%========================================================================
%-------------------------------- Functions
 ----------------------------
%========================================================================
% The function for a circular trajectory on the x-y plane based on
 states
function vals = refFun(theta)
    r = 3;
    val1 = r*cos(theta);
    val2 = r*sin(theta);
    vals = [val1;val2;3];
end

% State Transform function
function x = zTox(z,Phi_inv)
    x = Phi_inv*z;
end
function z = xToz(x,Phi)
    z = Phi*x;
end

% Controller function and helper functions
function u = FBL_Ctrl(Phi,Phi_inv,K,x,xr)
    sigma = 10;
    Del_inv = [1/sigma, 0; 0 1];
    zr = xToz(xr,Phi);
    z = xToz(x,Phi);

    beta_inv = [sigma, 0; 0 1];  % beta_inv = inv(Del_inv) = Del

    u = -beta_inv*(K*(z-zr) - alpha_fn(x));
```

```
    %u = beta_inv*(K*x - alpha_fn(x));

end

function alphax = alpha_fn(x)
    sigma = 10;
    Del_inv = [1/sigma, 0; 0 1];

    alphax = -Del_inv*b_fn(x);

end

function bx = b_fn(x)
    sigma = 10;
    rho = 28;
    beta = 8/3;
    bx1 = sigma^2*(x(1) - x(2)) + sigma*(x(1)*(rho - x(3)) - x(2));
    bx2 = x(1)*x(2) - beta*x(3);

    bx = [bx1;bx2];
end

% Drift Function
function f0 = f0_fn(x)
    sigma = 10;
    rho = 28;
    beta = 8/3;

    f01 = sigma*(x(2) - x(1));
    f02 = x(1)*(rho - x(3)) - x(2);
    f03 = x(1)*x(2) - beta*x(3);

    f0 = [f01;f02;f03];
end


Del =

[ s, 0]
[ 0, 1]


B =

 (x1 - x2)*s^2 + (x1*(r - x3) - x2)*s
                        x1*x2 - b*x3


alpha =

 ((x2 - x1)*s^2 + (x2 - x1*(r - x3))*s)/s
                            b*x3 - x1*x2
```

```
beta =

[ 1/s, 0]
[   0, 1]


Phi =

          x1
 -s*(x1 - x2)
          x3


dPhi =

[  1, 0, 0]
[ -s, s, 0]
[  0, 0, 1]


Phi_inv =

[ 1,   0, 0]
[ 1, 1/s, 0]
[ 0,   0, 1]


Az =

 -s*(x1 - x2)
            0
            0


B =

[ 0, 0]
[ 1, 0]
[ 0, 1]


Alqr =

     0     1     0
     0     0     0
     0     0     0


Blqr =

     0     0
     1     0
     0     1
```

```
Clqr =

     1     0     0
     0     1     0
     0     0     1


Qlqr =

    10     0     0
     0    10     0
     0     0    10


Rlqr =

     1     0
     0     1


Klqr =

    3.1623    4.0404         0
         0         0    3.1623
```



Published with MATLAB® R2017a