Transverse-bifurcation-of-viscous-slow-MHD-shocks (/github/rafael-a-monteiro-math/Transverse-bifurcation-of-viscous-slow-MHD-shocks/tree/7bec455020bcb32065a096dcea6d
/
BMZ-symbolic-computations.ipynb (/github/rafael-a-monteiro-math/Transverse-bifurcation-of-viscous-slow-MHD-shocks/tree/7bec455020bcb32065a096dcea6d8517660acd32/BM

This is a suplementary material to the paper Transverse bifurcation of viscous slow MHD shocks (https://arxiv.org/abs/1901.09153), by Blake Barker, Rafael Monteiro, and Kevin Zumbrun

Other relevant references are:

[FT] Freistuhler, Heinrich and Trakhinin, Yuri, *On the viscous and inviscid stability of magnetohydrodynamic shock waves,* Phys. D., Physica D. Nonlinear Phenomena, volume 237, number 23 (2008), 3030--3037.

[HLZ] J. Humpherys, G. Lyng, K. Zumbrun, *Multidimensional stability of large-amplitude Navier-Stokes shocks*, Archive for Rational Mechanics and Analysis 226.3 (2017): 923-973.

[Maj] A. Majda, *The stability of multi-dimensional shock fronts*, $Mem. Amer. Math. Soc.$, 41(275)(1983).

[MeZ] G. Metivier and K. Zumbrun, *Hyperbolic boundary value problems for symmetric systems with variable multiplicities*, J. of Differential Equations, 211 (2005) 61-134.

[ZS] K. Zumbrun and D. Serre, *Viscous And Inviscid Stability Of Multidimensional Planar Shock Fronts* Indiana Univ. Math. J, 48 (1999) 937-992.

[Hersh] Reuben Hersh, *Mixed Problems in Several Variables* Journal of Mathematics and Mechanics, Vol 12, No 3, 1963.

[Mont] Monteiro, Rafael, *Transverse steady bifurcation of viscous shock solutions of a system of parabolic conservation laws in a strip*, J. Differential Equations, Physica D. Nonlinear Phenomena, volume 257, number 6 (2014), 2035--2077.

[JYZ] Pogan, A. , Yao, J. and Zumbrun, K., *O(2) Hopf bifurcation of viscous shock waves in a channel*, Physica D: Nonlinear Phenomena 308 (2015): 59-79.

Mathematical setting

We want to analyse the stability of the shock wave solution to MHD equations, given below

$$\begin{aligned}
\rho_t + div(\rho u) =& 0\\
(\rho u)_t + div(\rho u \otimes u) + H \times curl(H) + \nabla p =& \nu \Delta u + (\lambda + \nu)\nabla div(u)\\
H_t + \nabla \times (H \times u) =& \eta \Delta H\\
\left(E + \frac{1}{2}H^2\right)_t + div((E+p)u + H \times (u \times H))) =& div(\sum U) + \kappa \Delta T + \eta(H \times (\nabla \times H)),
\end{aligned} \tag{1}$$

where $u \in \mathbb{R}^n$ is the fluid velocity, $\rho > 0$ is density, T is the temperature, $p = p(\rho, T)$ is pressure, $H \in \mathbb{R}^n$ is the magnetic field, $\sum$ is a viscous stress tensor and E is a given energy. $\lambda, \nu \geq 0$ are viscosity coefficients, $\kappa, \eta \geq 0$ are heat conductivity and electrical resistivity, respectively. \

Upon linearization (1) about a shock solution

$$\bar{U}(x - st) = U^{\pm} \quad , \quad \text{if} \quad x - st \leq 0$$

we obtain a system of equations that can be writen in the form

$$A_0^{\pm} U_t + A_1 U_x^{\pm} + A_2 U_y^{\pm} = 0 \tag{2}$$

where

$$A_0^{\pm} = \left(\begin{array}{c|c|c|c|c} 1 & 0 & 0 & 0 & 0 \\ u_1 & \rho & 0 & 0 & 0 \\ 0 & 0 & \rho & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right)^{\pm}, \quad A_1^{\pm} = \left(\begin{array}{c|c|c|c|c} u_1 & \rho & 0 & 0 & 0 \\ u_1^2 + p_\rho & 2\rho u_1 & 0 & 0 & 0 \\ 0 & 0 & \rho u_1 & 0 & -h_1 \\ 0 & 0 & 0 & \alpha & 0 \\ 0 & 0 & -h_1 & 0 & u_1 \end{array}\right)^{\pm}, \quad \text{and} \quad A_2^{\pm} = \left(\begin{array}{c|c|c|c|c} 0 & 0 & \rho & 0 & 0 \\ 0 & 0 & \rho u_1 & 0 & 0 \\ p_\rho & 0 & 0 & h_1 & 0 \\ 0 & 0 & h_1 & 0 & \alpha - u_1 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right)^{\pm}.$$

Laplace transforming (2) in time and Fourier transform it in the direction transversal to the direction of propagation we obtain

$$\lambda A_0^{\pm} \tilde{U} + A_1 U_x^{\pm} + i\xi A_2 \tilde{U} = 0$$

the problem now consists in solving the ODE

$$U_x^{\pm} = -(A_1^{\pm})^{-1}(\lambda A_0^{\pm} + i\xi A_2^{\pm})\tilde{U}. \tag{3}$$

Notice that this problem consists in two systems of ode's which are coupled through Rankine-Hugoniot conditions. Our goal is to understand the behavior of the solutions of (???). In order to do that we need to understand the spectrum and the eigenspaces of $-(A_1^{\pm})^{-1}(\lambda A_0^{\pm} + i\xi A_2^{\pm})$.

Simplifying the matrix $-(A_1^{\pm})^{-1}(\lambda A_0^{\pm} + i\xi A_2^{\pm})$.

Even though the mathematics tell us to invert a matrix, computationally it might be the case to avoid it or circumvent this operation. The more we can simplify the matrix $-(A_1^{\pm})^{-1}(\lambda A_0^{\pm} + i\xi A_2^{\pm})$ the better.

$$\lambda(A_1)^{-1}A_0 + i\xi(A_1)^{-1}A_2 = \left(\begin{array}{c|c|c|c|c} \frac{\lambda u_1}{u_1^2-p_\rho} & -\frac{\lambda\rho}{u_1^2-p_\rho} & \frac{i\,\rho u_1\xi}{u_1^2-p_\rho} & 0 & 0 \\ -\frac{\lambda p_\rho}{(u_1^2-p_\rho)\rho} & \frac{\lambda u_1}{u_1^2-p_\rho} & -\frac{i\,p_\rho\xi}{u_1^2-p_\rho} & 0 & 0 \\ \frac{i\,p_\rho u_1\xi}{\rho u_1^2-h_1^2} & 0 & \frac{\lambda\rho u_1}{\rho u_1^2-h_1^2} & \frac{i\,h_1u_1\xi}{\rho u_1^2-h_1^2} & \frac{h_1\lambda}{\rho u_1^2-h_1^2} \\ 0 & 0 & \frac{i\,h_1\xi}{\alpha} & \frac{\lambda}{\alpha} & \frac{(i\,\alpha - i\,u_1)\xi}{\alpha} \\ \frac{i\,h_1p_\rho\xi}{\rho u_1^2-h_1^2} & 0 & \frac{h_1\lambda\rho}{\rho u_1^2-h_1^2} & \frac{i\,h_1^2\xi}{\rho u_1^2-h_1^2} & \frac{\lambda\rho u_1}{\rho u_1^2-h_1^2} \end{array}\right) \tag{4}$$

As similar matrices have same spectrum, we will apply groups of operations $M \mapsto P \cdot M \cdot P^{-1}$ for specific matrices P in order to simplify our analysis as much as possible (since, in the end, we just want the spectrum of (2)).

Initially we set $S_1 := P_1(\lambda(A_1)^{-1}A_0 + i\xi(A_1)^{-1}A_2)P_1^{-1}$, where

$$P_1 = \left(\begin{array}{cc|c|cc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{h_1}{u_1} & 0 & 1 \end{array}\right)$$

We obtain

$$P_1(A_1)^{-1}(\lambda A_0 + i\xi A_2)P_1^{-1} = \left(\begin{array}{c|c|c|c|c} \frac{\lambda u_1}{u_1^2-p_\rho} & -\frac{\lambda\rho}{u_1^2-p_\rho} & \frac{i\,\rho u_1\xi}{u_1^2-p_\rho} & 0 & 0 \\ -\frac{\lambda p_\rho}{(u_1^2-p_\rho)\rho} & \frac{\lambda u_1}{u_1^2-p_\rho} & -\frac{i\,p_\rho\xi}{u_1^2-p_\rho} & 0 & 0 \\ \frac{i\,p_\rho u_1\xi}{\rho u_1^2-h_1^2} & 0 & \frac{(\rho u_1^2+h_1^2)\lambda}{(\rho u_1^2-h_1^2)u_1} & \frac{i\,h_1u_1\xi}{\rho u_1^2-h_1^2} & \frac{h_1\lambda}{\rho u_1^2-h_1^2} \\ 0 & 0 & \frac{i\,h_1\xi}{u_1} & \frac{\lambda}{\alpha} & \frac{(i\,\alpha - i\,u_1)\xi}{\alpha} \\ 0 & 0 & \frac{h_1\lambda}{u_1^2} & 0 & \frac{\lambda}{u_1} \end{array}\right)$$

Now, using the matrix

$$P_2 = \left(\begin{array}{c|c|c|c|c} 1 & 0 & 0 & 0 & 0 \\ \frac{p_\rho}{\rho u_1} & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right)$$

We set $S_2 := P_2 \cdot S_1 \cdot P_2^{-1}$ to get

$$S_2 = \left(\begin{array}{c|c|c|c|c} \frac{\lambda(u_1^2+p_\rho)}{u_1(u_1^2-p_\rho)} & -\frac{\lambda\rho}{u_1^2-p_\rho} & \frac{i\,\rho u_1\xi}{u_1^2-p_\rho} & 0 & 0 \\ -\frac{p_\rho\lambda}{\rho u_1^2} & \frac{\lambda}{u_1} & 0 & 0 & 0 \\ \frac{i\,p_\rho u_1\xi}{\rho u_1^2-h_1^2} & 0 & \frac{(\rho u_1^2+h_1^2)\lambda}{(\rho u_1^2-h_1^2)u_1} & \frac{i\,h_1u_1\xi}{\rho u_1^2-h_1^2} & \frac{h_1\lambda}{\rho u_1^2-h_1^2} \\ 0 & 0 & \frac{i\,h_1\xi}{u_1} & \frac{\lambda}{\alpha} & \frac{(i\,\alpha - i\,u_1)\xi}{\alpha} \\ 0 & 0 & \frac{h_1\lambda}{u_1^2} & 0 & \frac{\lambda}{u_1} \end{array}\right)$$

We multiply then by the diagonal matrix $D = (1, 1, h_1/u_1, 1, 1)$:

$$S_3 := DS_2D^{-1} = \left(\begin{array}{c|c|c|c|c} \frac{\lambda(u_1^2+p_\rho)}{u_1(u_1^2-p_\rho)} & -\frac{\lambda\rho}{u_1^2-p_\rho} & \frac{i\,\rho u_1^2\xi}{h_1(u_1^2-p_\rho)} & 0 & 0 \\ -\frac{p_\rho\lambda}{\rho u_1^2} & \frac{\lambda}{u_1} & 0 & 0 & 0 \\ \frac{i\,p_\rho h_1\xi}{\rho u_1^2-h_1^2} & 0 & \frac{(\rho u_1^2+h_1^2)\lambda}{(\rho u_1^2-h_1^2)u_1} & \frac{i\,h_1^2\xi}{\rho u_1^2-h_1^2} & \frac{h_1^2\lambda}{u_1(\rho u_1^2-h_1^2)} \\ 0 & 0 & i\xi & \frac{\lambda}{\alpha} & \frac{i(\alpha-u_1)\xi}{\alpha} \\ 0 & 0 & \frac{\lambda}{u_1} & 0 & \frac{\lambda}{u_1} \end{array}\right)$$

The matrix above is similar to $A_1^{-1}(\lambda A_0 + i\xi A_2)$, hence $-A_1^{-1}(\lambda A_0 + i\xi A_2)$ is similar to $S = -S_3$, which is

$$S = \left(\begin{array}{c|c|c|c|c} -\frac{\lambda(u_1^2+p_\rho)}{u_1(u_1^2-p_\rho)} & \frac{\lambda\rho}{u_1^2-p_\rho} & -\frac{i\,\rho u_1^2\xi}{(u_1^2-p_\rho)h_1} & 0 & 0 \\ \frac{p_\rho\lambda}{\rho u_1^2} & -\frac{\lambda}{u_1} & 0 & 0 & 0 \\ -\frac{i\,h_1p_\rho\xi}{\rho u_1^2-h_1^2} & 0 & -\frac{\lambda(\rho u_1^2+h_1^2)}{(\rho u_1^2-h_1^2)u_1} & -\frac{i\,h_1^2\xi}{\rho u_1^2-h_1^2} & -\frac{h_1^2\lambda}{(\rho u_1^2-h_1^2)u_1} \\ 0 & 0 & -i\xi & -\frac{\lambda}{\alpha} & -\frac{i\,(\alpha-u_1)\xi}{\alpha} \\ 0 & 0 & -\frac{\lambda}{u_1} & 0 & -\frac{\lambda}{u_1} \end{array}\right)$$

We display the code for this part below.

In [2]:
```
var('alpha rho u_1 p_rho h_1 xi lambd lambd_0 lambd_1 lambd_2 mu_0 mu_1 mu_2')

A_0 = matrix([[1,0,0, 0,0],[u_1,rho, 0,0,0], \
              [ 0,0,rho, 0,0], [0,0,0,1,0], [0,0,0,0,1]]);
A_1 = matrix([[u_1,rho,0, 0,0],[u_1*u_1 + p_rho,2*rho*u_1,0,0,0], \
              [ 0,0,rho*u_1, 0,-h_1], [0,0,0,alpha,0], [0,0,-h_1,0,u_1]]);
A_2 = matrix([[0,0,rho, 0,0],[0,0,rho*u_1,0,0], [ p_rho,0,0, h_1,0],\
              [0,0,h_1,0,alpha - u_1], [0,0,0,0,0]]);

### Inverse of A_1
A_1_inv_aux = matrix([[2*rho*u_1,-rho,0, 0,0],\
                      [-(u_1*u_1 + p_rho), u_1,0,0,0], [ 0,0,u_1, 0,h_1], [0,0,0,1/alpha,0], [0,0,h_1,0,rho*u_1]]);
a_1= 1/(rho*(u_1^2 - p_rho));
a_2 =1/(rho*u_1^2 - h_1^2);

D = diagonal_matrix([a_1,a_1, a_2,1,a_2]);
A_1_inv = D*A_1_inv_aux;

##############################################

S = lambd*A_1_inv*A_0 + i*xi*A_1_inv*A_2;
P_1 = matrix([[1,0,0,0,0],[0,1,0,0,0],[0,0,1,0,0],[0,0,0,1,0],[0,0,-h_1/u_1,0,1]]);

####Now we do P_1*S*P_1.inverse() , at equation (7)
S = P_1*S*P_1.inverse();

### Simplifying the second row
P_2 = matrix([[1,0,0,0,0],[p_rho/(rho*u_1),1,0,0,0],[0,0,1,0,0],[0,0,0,1,0],[0,0,0,0,1]]);

### ....we obtain equation (9)
S = P_2*S*P_2.inverse();

### Then we do S = P_2*S*S_2.inverse()
D = diagonal_matrix([1, 1, h_1/u_1, 1, 1]);

### Equation (10)
S = D*S*D.inverse();
S = -S;
S = S.expand();
```

A brief digression: on the dynamics of doing symbolic computations

Even though SAGE has some routines implemented, asking for it to calculate the spectrum of $-A_1^{-1} * S$ can cost you hours. Even worse: it can provide you hundreds of lines of solution, which are useless for analytical studies. As in the parallel case, the idea is to simplify. First of all, it is much easier to compute (symbolically) the mapping

$$\mu \quad \mapsto \quad det(\mu \cdot A_1 + S)$$

than computing $\mu \mapsto det(\mu \cdot I + A_1^{-1} \cdot S)$. As before, we explore the fact that similar matrices have the same spectrum to rewrite $\mu \cdot A_1 + S$ as

$$\begin{pmatrix} \mu u_1 + \lambda + i\,u_2 & \mu\rho & i\,\rho & 0 & 0 \\ 0 & \mu\rho u_1 + \lambda\rho + i\,\rho u_2 & i\,\mu^2\rho u_1 + i\,\lambda\mu\rho - \mu\rho u_2 & -i\,h_2\mu^2 - h_1\mu - i\,h_2 & -i\,h_1\mu^2 + 2\,h_2\mu \\ i\,p_\rho & 0 & \mu\rho u_1 + \lambda\rho + i\,\rho u_2 & -h_2\mu + i\,h_1 & -h_1\mu - i\,h_2 \\ 0 & -i\,h_2 & i\,h_1 & \alpha\mu + \lambda + i\,u_2 & i\,\alpha - i\,u_1 \\ 0 & 0 & 0 & -i\,\alpha\mu^2 - i\,\lambda\mu & \alpha\mu + \lambda \end{pmatrix} \tag{5}$$

And now we need to analyse case by case.

> Remark 1: it is not hard to conclude by inspectioning the last row of the matrix in (5) that $\mu = -\frac{\lambda}{\alpha}$ is an eigenvalue of $-A_1^{-1} \cdot S$.

The associated eigenvectors

We finally show how to derive an explicit formula for the eigenvectors associated to the matrix (7) withouth knowing the eigenvalues. For the next results we will make use of the following observation, proved by inspection using the result on this section.

> Observation 2: None of the eigenvalues of the matrix S in (7) is a diagonal entry of S,except for $-\frac{\lambda}{\alpha}$.

We begin proving the following lemma:

Lemma 3: Let $\mu \in \sigma(S) \setminus \{-\frac{\lambda}{\alpha}\}$ and x be its associated eigenvector.Let $\lambda \neq 0$. Then, $x = (x_1, \dots, x_5)$ has no null entries.

Proof: Since x is an eigenvector associated to μ then x is in the kernel of $S - \mu I$, i.e.,

$$(S-\mu I)x = \begin{pmatrix} -\frac{\lambda(u_1^2+p_\rho)}{u_1(u_1^2-p_\rho)} - \mu & \frac{\lambda\rho}{u_1^2-p_\rho} & -\frac{i\,\rho u_1^2\xi}{(u_1^2-p_\rho)h_1} & 0 & 0 \\ \frac{p_\rho\lambda}{\rho u_1^2} & -\frac{\lambda}{u_1} - \mu & 0 & 0 & 0 \\ -\frac{i\,h_1 p_\rho \xi}{\rho u_1^2 - h_1^2} & 0 & -\frac{\lambda(\rho u_1^2+h_1^2)}{(\rho u_1^2-h_1^2)u_1} - \mu & -\frac{i\,h_1^2\xi}{\rho u_1^2-h_1^2} & -\frac{h_1^2\lambda}{(\rho u_1^2-h_1^2)u_1} \\ 0 & 0 & -i\xi & -\frac{\lambda}{\alpha} - \mu & -\frac{i\,(\alpha-u_1)\xi}{\alpha} \\ 0 & 0 & -\frac{\lambda}{u_1} & 0 & -\frac{\lambda}{u_1} - \mu \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = 0. \tag{6}$$

Notice that the assumption implies that none of the diagonal entries of the above matrix is zero. The proof goes by contradiction: assume that x has a null entry. Now we have to analyse each one of the 5 possible cases:

a) $x_1 = 0 \overset{2^{nd}\,equation}{\Longrightarrow} x_2 = 0 \overset{1^{st}\,equation}{\Longrightarrow} x_3 = 0 \overset{5^{th}\,equation}{\Longrightarrow} x_5 = 0 \overset{3^{rd}\,equation}{\Longrightarrow} x_4 = 0$. Therefore, $x = 0$ (absurd!).

b) $x_2 = 0 \overset{2^{nd}\,equation}{\Longrightarrow} x_1 = 0 \overset{1^{st}\,equation}{\Longrightarrow} x_3 = 0 \overset{5^{th}\,equation}{\Longrightarrow} x_5 = 0 \overset{4^{th}\,equation}{\Longrightarrow} x_4 = 0$. Therefore, $x = 0$ (absurd!).

c) $x_3 = 0 \overset{5^{th}\,equation}{\Longrightarrow} x_5 = 0 \overset{4^{th}\,equation}{\Longrightarrow} x_4 = 0 \overset{3^{rd}\,equation}{\Longrightarrow} x_1 = 0 \overset{2^{nd}\,equation}{\Longrightarrow} x_2 = 0$. Therefore, $x = 0$ (absurd!).

d) $x_4 = 0$. As $\mu \neq \frac{\lambda}{\alpha}$, the last two rows of (6) are linearly independent; thus $x_3 = x_5 = 0$. Now, use third equation to get $x_1 = 0 \overset{2^{nd}\,equation}{\Longrightarrow} x_2 = 0$. Therefore, $x = 0$ (absurd!).

e) $x_5 = 0 \overset{5^{th}\,equation}{\Longrightarrow} x_3 = 0 \overset{4^{th}\,equation}{\Longrightarrow} x_4 = 0 \overset{3^{rd}\,equation}{\Longrightarrow} x_1 = 0 \overset{2^{nd}\,equation}{\Longrightarrow} x_2 = 0$. Therefore, $x = 0$ (absurd!). ■

Corollary 4: Let $\mu \in \sigma(S) \setminus \{-\frac{\lambda}{\alpha}\}$ and x be its associated eigenvector. Then, x is given by the following formula:

$$x = x(\mu) = \begin{pmatrix} \frac{-i\,\mu^2\rho u_1^3 - 2i\,\lambda\mu\rho u_1^2 - \left(-i\,h_1^2\mu^2 + i\,\lambda^2\rho + i\,h_1^2\right)u_1}{h_1\lambda p_\rho} \\ \frac{-i\,\mu^2\rho u_1^2 + i\,h_1^2\mu^2 - 2i\,\lambda\mu\rho u_1 - i\,\lambda^2\rho - i\,h_1^2}{h_1\mu\rho u_1 + h_1\lambda\rho} \\ -\frac{\mu u_1 + \lambda}{\lambda} \\ \frac{i\,u_1}{\lambda} \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{-i\rho u_1(\mu u_1+\lambda)^2 + i u_1 h_1^2(\mu^2-1)}{h_1\lambda p_\rho} \\ \frac{-i\rho(\mu u_1+\lambda)^2 + i h_1^2(\mu^2-1)}{h_1\rho(\mu u_1+\lambda)} \\ -\frac{(\mu u_1+\lambda)}{\lambda} \\ \frac{i u_1}{\lambda} \\ 1 \end{pmatrix} \tag{7}$$

Proof: By Lemma 3 we can scale x by any of its entries. W.l.og., we set $x_5 = 1$ and use equation (6) to find the other entries. ■

Remark 5: Notice that only entries that are purely imaginary in (7) are the first, second and fourth, as in FT, Eq. (60) .

Remark 6: We can rescale the vector \eqref{eigenvector_depending_on*eigenvalue} multiplying by ${\left(\mu u{1} + \lambda\right)} \lambda/h_1 *and write* x(\mu)$ as \begin{equation}\tag{8}\label{eigenvector} x(\mu) \overset{\epsilon = 1/h_1}{=} \left(\begin{array}{c} \frac{i\,u_1 \left(\mu u_{1} + \lambda\right) (\mu^2 -1)}{p_{\rho}}\\ \frac{i\lambda(\mu^2 -1)}{\rho} \\ 0\\ 0\\ 0 \end{array} \right) +\epsilon \left(\begin{array}{c} 0\ 0\

- (\mu u*{1} + \lambda)^{2} \ i \,u{1} {\left(\mu u*{1} + \lambda\right)} \ {\left(\mu u{1} + \lambda\right)} \lambda \end{array} \right)+ \epsilon^2 \left(

$$\begin{array}{c} \frac{-i\rho u_1(\mu u_1+\lambda)^3}{p_\rho} \\ -i\lambda(\mu u_1+\lambda)^2 \\ 0 \\ 0 \\ 0 \end{array}$$

\right) \end{equation}

In [4]:
```
var('x_1 x_2 x_3 x_4 x_5 num ')
var('alpha rho u_1 p_rho h_1 xi lambd lambd_0 lambd_1 lambd_2 mu_0 mu_1 mu_2 mu epsilon ')

A = S - mu*identity_matrix(5);

A = A.substitute(alpha = u_1);
x = column_matrix([x_1,x_2, x_3, x_4,lambd/h_1]);
f(x_1,x_2, x_3, x_4,x_5)= (A[(4),(0,1,2,3,4,)]*x())[0,0];
sols = solve( [f()== 0], x_3);
a= x;
a = a.subs(sols[0]);

### x_4 must be....
f(x_1,x_2, x_3, x_4)= (A[(3),(0,1,2,3,4,)]*x())[0,0];
sols= solve( [f(x_3= a[2][0])== 0 ], x_4);
a = a.subs(sols[0]);

### x_1 must be....
f(x_1,x_2, x_3, x_4)= (A[(2),(0,1,2,3,4,)]*x())[0,0];
sols= solve( [f(x_3 = a[2][0],x_4= a[3][0])== 0], x_1);
a = a.subs(sols[0]);
a = a.substitute((-I*mu^2*rho*u_1^3 - I*h_1^2*u_1*xi^2 \
                 - 2*I*lambd*mu*rho*u_1^2 - (-I*h_1^2*mu^2 + I*lambd^2*rho)*u_1)/(h_1^2*p_rho*xi)\
                == ( -i*rho*u_1*(mu*u_1 + lambd)^2/(h_1^2*p_rho*xi) + i*u_1*(mu^2 - xi^2)/(p_rho*xi)));   ##OK

#### and finally, x_2 must be....
f(x_1,x_2, x_3, x_4)= (A[(1),(0,1,2,3,4,)]*x())[0,0];
sols= solve( [f(x_3 = a[2][0],x_5= a[4][0],x_1= a[0][0],x_4= a[3][0] )== 0], x_2);

x_mu= a.subs(sols[0]);
#x_mu[1] = x_mu[1].substitute((h_1*mu*rho*u_1 + h_1*lambd*rho)*xi ==h_1*rho*(mu*u_1 +lambd)*xi )
x_mu[1] = x_mu[1].substitute(((-I*mu^2*rho*u_1^2 + I*h_1^2*mu^2 - I*h_1^2*xi^2 - 2*I*lambd*mu*rho*u_1 - I*lambd^2*rho)*la
mbd/((h_1*mu*rho*u_1 + h_1*lambd*rho)*xi)) == lambd*(-i*rho*(mu*u_1 +lambd)^2 + i*h_1^2*(mu^2 -xi^2))/(h_1*rho*xi*(mu*u_1
 + lambd))); ##OK
x_mu = x_mu.substitute(((-I*lambd*mu^2*rho*u_1^2 + I*h_1^2*lambd*mu^2 - I*h_1^2*lambd*xi^2 - 2*I*lambd^2*mu*rho*u_1 - I*l
ambd^3*rho)/((h_1^2*mu*rho*u_1 + h_1^2*lambd*rho)*xi)) == (-I*lambd*rho*(mu*u_1+ lambd)/(h_1^2*rho*xi)  + I*lambd*(mu^2 -
xi^2)/(rho*xi*(mu*u_1+ lambd)) )) ; ##OK

x_mu = (mu*u_1+lambd)*x_mu;

x_mu[1] = -I*(mu*u_1 + lambd)^2*lambd/(h_1^2*xi) + I*(mu^2 - xi^2)*lambd/(rho*xi);   #### INTERVENTION

x_mu = x_mu.substitute(h_1 = 1/epsilon,xi=1);
```

Defining $\epsilon = \frac{1}{h_1}$, note that

$$\frac{1}{a^2 - h_1^2} \overset{h_1 \to \infty}{=} -\frac{1}{h_1^2}\left(1 + \left(\frac{a}{h_1}\right)^2 + \dots\right)$$

holds, hence we can rewrite the matrix S in (9) as

$$S = S(\epsilon) \approx \left(\begin{array}{ccc|c|c} -\frac{\lambda(u_1^2+p_\rho)}{u_1(u_1^2-p_\rho)} & \frac{\lambda\rho}{u_1^2-p_\rho} & -\frac{i\,\rho u_1^2 \xi}{(u_1^2-p_\rho)}\epsilon & 0 & 0 \\ \frac{p_\rho \lambda}{\rho u_1^2} & -\frac{\lambda}{u_1} & 0 & 0 & 0 \\ i\,p_\rho \xi\epsilon\left(1+\rho\epsilon^2 u_1^2\right) & 0 & \frac{\lambda}{u_1} + 2\rho u_1 \lambda\epsilon^2 & i\xi + iu_1^2\rho\xi\epsilon^2 & \frac{\lambda}{u_1} + \rho u_1 \lambda\epsilon^2 \\ 0 & 0 & -i\xi & -\frac{\lambda}{\alpha} & -\frac{i\,(\alpha-u_1)\xi}{\alpha} \\ 0 & 0 & -\frac{\lambda}{u_1} & 0 & -\frac{\lambda}{u_1} \end{array}\right)$$

Observation 8: It is not hard to see that $-\frac{\lambda}{\alpha}$ is an eigenvalue of S.

In [3]:
```
S = S.substitute(h_1=1/epsilon,xi=1);
S = S.substitute( rho*u_1^2 - 1/epsilon^2 == (rho*u_1^2*epsilon^2 -1)/epsilon^2  );     ##OK
S = S.substitute( -epsilon^2*lambd*rho*u_1/(epsilon^2*rho*u_1^2 - 1) - lambd/((epsilon^2*rho*u_1^2 - 1)*u_1)\
                == -lambd*(epsilon^2*rho*u_1^2 + 1)/((epsilon^2*rho*u_1^2 - 1)*u_1));   ##OK
S = S.substitute( -epsilon^2*lambd*rho*u_1/(epsilon^2*rho*u_1^2 - 1) + lambd/((epsilon^2*rho*u_1^2 - 1)*u_1)\
                == -lambd/u_1);   ##OK
S = S.substitute(         lambd*p_rho/((u_1^2 - p_rho)*rho) - lambd*p_rho^2/((u_1^2 - p_rho)*rho*u_1^2)\
                == lambd*p_rho/(rho*u_1^2));   ##OK
S= S.substitute(-lambd*u_1/(u_1^2 - p_rho) + lambd*p_rho/((u_1^2 - p_rho)*u_1) == -lambd/u_1);         ##OK
S= S.substitute(-lambd*u_1/(u_1^2 - p_rho) - lambd*p_rho/((u_1^2 - p_rho)*u_1) \
                == -lambd*(u_1^2 + p_rho)/((u_1^2 - p_rho)*u_1));     ##OK
```

After a huge computation like this one we can easily print the result:

In [4]:
```
S
```

Out[4]:
```
[                    -(u_1^2 + p_rho)*lambd/((u_1^2 - p_rho)*u_1)                                                  lambd*rho
/(u_1^2 - p_rho)                                -I*epsilon*rho*u_1^2/(u_1^2 - p_rho)
                                 0                                                                                0]
[                                           lambd*p_rho/(rho*u_1^2)
      -lambd/u_1                                                                  0
                                 0                                                                                0]
[                       -I*epsilon*p_rho/(epsilon^2*rho*u_1^2 - 1)
               0 -(epsilon^2*rho*u_1^2 + 1)*lambd/((epsilon^2*rho*u_1^2 - 1)*u_1)
    -I/(epsilon^2*rho*u_1^2 - 1)                                -lambd/((epsilon^2*rho*u_1^2 - 1)*u_1)]
[                                                                    0
                                                                                 -I
               0                                                                    I*u_1/alpha - I]
                    -lambd/alpha
[                                                                    0
                                                                           -lambd/u_1
               0                                                                        -lambd/u_1]
                                 0
```

and also in latex format$

In [5]:
```
latex(S)
```

Out[5]:
```
\left(\begin{array}{rrrrr}
-\frac{{\left(u_{1}^{2} + p_{\rho}\right)} \mathit{lambd}}{{\left(u_{1}^{2} - p_{\rho}\right)} u_{1}} & \frac{\mat
hit{lambd} \rho}{u_{1}^{2} - p_{\rho}} & -\frac{i \, \epsilon \rho u_{1}^{2}}{u_{1}^{2} - p_{\rho}} & 0 & 0 \\
\frac{\mathit{lambd} p_{\rho}}{\rho u_{1}^{2}} & -\frac{\mathit{lambd}}{u_{1}} & 0 & 0 & 0 \\
-\frac{i \, \epsilon p_{\rho}}{\epsilon^{2} \rho u_{1}^{2} - 1} & 0 & -\frac{{\left(\epsilon^{2} \rho u_{1}^{2} +
1\right)} \mathit{lambd}}{{\left(\epsilon^{2} \rho u_{1}^{2} - 1\right)} u_{1}} & -\frac{i}{\epsilon^{2} \rho u_{1
}^{2} - 1} & -\frac{\mathit{lambd}}{{\left(\epsilon^{2} \rho u_{1}^{2} - 1\right)} u_{1}} \\
0 & 0 & -i & -\frac{\mathit{lambd}}{\alpha} & \frac{i \, u_{1}}{\alpha} - i \\
0 & 0 & -\frac{\mathit{lambd}}{u_{1}} & 0 & -\frac{\mathit{lambd}}{u_{1}}
\end{array}\right)
```

$$\begin{pmatrix} -\frac{(u_1^2+p_\rho)lambd}{(u_1^2-p_\rho)u_1} & \frac{lambd\rho}{u_1^2-p_\rho} & -\frac{i\,\rho u_1^2}{(u_1^2-p_\rho)h_1} & 0 & 0 \\ \frac{lambdp_\rho}{\rho u_1^2} & -\frac{lambd}{u_1} & 0 & 0 & 0 \\ -\frac{i\,h_1 p_\rho}{\rho u_1^2-h_1^2} & 0 & -\frac{(\rho u_1^2+h_1^2)lambd}{(\rho u_1^2-h_1^2)u_1} & -\frac{i\,h_1^2}{\rho u_1^2-h_1^2} & -\frac{h_1^2 lambd}{(\rho u_1^2-h_1^2)u_1} \\ 0 & 0 & -i & -\frac{lambd}{u_1} & 0 \\ 0 & 0 & -\frac{lambd}{u_1} & 0 & -\frac{lambd}{u_1} \end{pmatrix}$$

Eigenvalues

In the case of an inviscid shock, we have two endstates accross a jump. Thus, the matrix S in (7) has upperscripts $\pm$ denoting the endstates at $\pm\infty$. Notice that, upon scaling, we can take $\xi = 1$ (redefine $\lambda \to \lambda\xi$). Assuming that

$$\lambda = \lambda_0 + \epsilon\lambda_1 + \epsilon^2\lambda_2 + \mathcal{O}(\epsilon^3)$$

we can find the spectrum of $S^{\pm}$ in terms of ϵ; moreover, the characteristic polynomial $p(x) = det(xI - S(\lambda_0 + \epsilon\lambda_1 + \ldots) = p_0(x) + \epsilon p_1(x) + \epsilon^2 p_2(x) + \ldots$ of S can be computed explicitly using SAGE. We consider then an asymptotic expansion for the eigenvalues x of the formulas

$$x = x_0 + \epsilon x_1 + \epsilon^2 x_2 \ldots .$$

For the sake of notation we drop the indexes "$\pm$" for now, since these formulas are the same on both sides. Last, as pointed out in Remark 1, one can easily check that $p(-\frac{\lambda}{\alpha}) = 0$.

Eigenvalue expansion in terms of ϵ

An easy calculation shows that the roots of the characteristic polynomial associated to (7), besides $-\frac{\lambda}{\alpha}$, are[1]

$$x_0 \quad \in \quad \left\{ \quad -\frac{\lambda_0}{u_1 \pm \sqrt{p_\rho}} \quad , \quad \pm 1 \quad \right\}$$

Higher order terms in the expansion of the eigenvalues in terms of ϵ can be easily obtained using SAGE, because once we have $x_0 \ \ldots \ x_i$ the problem of finding x_{i+1} is linear. They can be written explicitly, but the computations are cumbersome: to avoid human error, we coded it in SAGE (see next cell).

> Remark 9 (positivity): As we are looking for instability, we must have $\lambda > 0$. Considering the expansion
> $$\lambda = \lambda_0 + \lambda_1\epsilon \ldots$$
> it suffices to find and index j such that $\lambda_j > 0$ and $\lambda_i = 0 \, \forall \, i < j$.

In [6]:
```
var('x x_0 x_1 x_2 x_3 epsilon rho u_1 p_rho alpha M  lambd lambda_0 lambda_1 lambda_2')

var('u_1_plus p_rho_plus rho_plus')       #"plus" variables

var('u_1_minus p_rho_minus rho_minus ')   #"minus" variables

###------------------------------------------------------------------------------

aux= S[2,0];
aux= aux.taylor(epsilon,0,3);
S[2,0] = aux;

aux= S[2,2];
aux= aux.taylor(epsilon,0,3);
S[2,2] = aux;

aux= S[2,3];
aux= aux.taylor(epsilon,0,3);
S[2,3] = aux;

aux= S[2,4];
aux= aux.taylor(epsilon,0,3);
S[2,4] = aux;

S = S.expand();

p(x,epsilon) = S.characteristic_polynomial();
g(x,epsilon,lambd) = (x + lambd/alpha);
h(x,epsilon,lambd) = (p.maxima_methods().divide(g))[0] ;

p(x,epsilon) = h.substitute(lambd = lambda_0 + epsilon*lambda_1 + epsilon^2*lambda_2);

p = p.expand();

q(epsilon) = p.substitute(x = x_0+epsilon*x_1+ epsilon^2*x_2 + epsilon^3*x_3);

q = q.expand();   ### We expand the characteristic polynomial in epsilonsilon terms ...

### and then we collect each term
Q_0(x_0,x_1,x_2,x_3) = q.coefficient(epsilon,0);
Q_1(x_0,x_1,x_2,x_3) = q.coefficient(epsilon,1);
Q_2(x_0,x_1,x_2,x_3) = q.coefficient(epsilon,2);
Q_3(x_0,x_1,x_2,x_3) = q.coefficient(epsilon,3);

### This is a matrix so that each row has the expansion o an eigenvalue up to order three.
### for instance, the first row corresponds to the eigenvalue
### lambda = roots[0,0] + epsilon*roots[0,1] + epsilon^2*roots[0,2] +...

roots = S[(0,1,2,3),(0,1,2,3)];
roots = 0*roots;
roots[0,0] = Q_0.roots(x_0)[0][0];
```

```
roots[1,0] = Q_0.roots(x_0)[1][0];
roots[2,0] = Q_0.roots(x_0)[2][0];
roots[3,0] = Q_0.roots(x_0)[3][0];

## Simplifying things a bit
roots = \
roots.substitute( -(lambda_0*u_1 - lambda_0*sqrt(p_rho))/(u_1^2 - p_rho) == -lambda_0/(u_1 + sqrt(p_rho))); ##OK
roots = \
roots.substitute(-(lambda_0*u_1 + lambda_0*sqrt(p_rho))/(u_1^2 - p_rho) == -lambda_0/(u_1 - sqrt(p_rho)));  ##OK

#########################################################################
### Then we proceed as follows

#-----------------------------------------
# x_0 = roots[0,0] = -lambda_0/(u_1 + sqrt(p_rho))
#-----------------------------------------

roots[0,1]= x_1;
aux = solve([Q_1(x_0 = roots[0,0])==0],x_1);
roots = roots.subs(aux[0]);
roots[0,1] = roots[0,1].substitute((lambda_0^2*lambda_1*sqrt(p_rho) - lambda_1*sqrt(p_rho)*u_1^2 \
                                    - 2*lambda_1*p_rho*u_1 - lambda_1*p_rho^(3/2))\
                                   /(sqrt(p_rho)*u_1^3 - lambda_0^2*p_rho + 3*p_rho*u_1^2 \
                                     - (lambda_0^2*sqrt(p_rho) - 3*p_rho^(3/2))*u_1 + p_rho^2) \
                                   == -lambda_1/(u_1 +sqrt(p_rho))); ##OK

roots[0,2]= x_2;
aux = solve([Q_2(x_0= roots[0,0],x_1=roots[0,1])==0],x_2);
roots = roots.subs(aux[0]);
roots[0,2] = roots[0,2]\
.substitute(-1/2*(lambda_0*p_rho^3*rho + 2*lambda_2*sqrt(p_rho)*u_1^3 - 2*lambda_0^2*lambda_2*p_rho \
                 + (lambda_0*p_rho^2*rho + 6*lambda_2*p_rho)*u_1^2 + 2*lambda_2*p_rho^2 \
                 + 2*(lambda_0*p_rho^(5/2)*rho - lambda_0^2*lambda_2*sqrt(p_rho) \
                      + 3*lambda_2*p_rho^(3/2))*u_1)/(sqrt(p_rho)*u_1^4 + 4*p_rho*u_1^3 - lambda_0^2*p_rho^(3/2) \
                                                      - (lambda_0^2*sqrt(p_rho) - 6*p_rho^(3/2))*u_1^2 \
                                                      + p_rho^(5/2) - 2*(lambda_0^2*p_rho - 2*p_rho^2)*u_1) \
            ==- lambda_2/(u_1 + sqrt(p_rho)) + lambda_0*sqrt(p_rho)^3*rho/(2*(lambda_0^2 - (u_1 + sqrt(p_rho))^2))); ##OK

roots[(0),(0,1,2,3)] = roots[(0),(0,1,2,3)].substitute(rho = rho_plus , u_1 \
                                                       = u_1_plus, p_rho = p_rho_plus, xi=1);

#-----------------------------------------
# x_0 = roots[1,0] =  - lambda_0/(u_1 - sqrt(p_rho))
#-----------------------------------------

roots[1,1]= x_1;
aux = solve([Q_1(x_0 = roots[1,0])==0],x_1);
roots = roots.subs(aux[0]);
roots[1,1] =roots[1,1].substitute(  (lambda_0^2*lambda_1*sqrt(p_rho) - lambda_1*sqrt(p_rho)*u_1^2 + 2*lambda_1*p_rho*u_1
- lambda_1*p_rho^(3/2))/(sqrt(p_rho)*u_1^3 + lambda_0^2*p_rho - 3*p_rho*u_1^2 - (lambda_0^2*sqrt(p_rho) - 3*p_rho^(3/2))*
u_1 - p_rho^2) ==
                                      -lambda_1/(u_1 -sqrt(p_rho))); ##OK

roots[1,2]= x_2;
aux = solve([Q_2(x_0= roots[1,0],x_1=roots[1,1])==0],x_2);
roots = roots.subs(aux[0]);
roots[1,2] = roots[1,2].substitute(1/2*(lambda_0*p_rho^3*rho - 2*lambda_2*sqrt(p_rho)*u_1^3 - 2*lambda_0^2*lambda_2*p_rho
 + (lambda_0*p_rho^2*rho + 6*lambda_2*p_rho)*u_1^2 + 2*lambda_2*p_rho^2 - 2*(lambda_0*p_rho^(5/2)*rho - lambda_0^2*lambda
_2*sqrt(p_rho) + 3*lambda_2*p_rho^(3/2))*u_1)/(sqrt(p_rho)*u_1^4 - 4*p_rho*u_1^3 - lambda_0^2*p_rho^(3/2) - (lambda_0^2*s
qrt(p_rho) - 6*p_rho^(3/2))*u_1^2 + p_rho^(5/2) + 2*(lambda_0^2*p_rho - 2*p_rho^2)*u_1)
                                    == -lambda_2/(u_1 -sqrt(p_rho)) - rho*lambda_0*sqrt(p_rho)^3/(2*(lambda_0^2 - (u_1 -sq
rt(p_rho))^2))); ##OK

roots[(1),(0,1,2,3)] = roots[(1),(0,1,2,3)].substitute(rho = rho_plus , u_1 \
                                                       = u_1_plus, p_rho = p_rho_plus, xi=1);

#-----------------------------------------
#x_0 = roots[2,0] =  1
#-----------------------------------------

roots[2,1]= x_1;
aux = solve([Q_1(x_0 = roots[2,0])==0],x_1)
roots = roots.subs(aux[0]);

roots[2,2]= x_2;
aux = solve([Q_2(x_0= roots[2,0],x_1=roots[2,1])==0],x_2);
roots = roots.subs(aux[0]);
roots[2,2] = \
roots[2,2].substitute( 1/2*(lambda_0^4*rho \
                            + 4*lambda_0^3*rho*u_1 + 6*lambda_0^2*rho*u_1^2 + 4*lambda_0*rho*u_1^3 + rho*u_1^4)\
                      /(lambda_0^2 + 2*lambda_0*u_1 + u_1^2 - p_rho) \
                      == rho*(lambda_0 +u_1)^4/(2*((lambda_0 +u_1)^2 - p_rho))); ##OK

roots[2,3]= x_3;
aux = solve([Q_3(x_0= roots[2,0],x_1=roots[2,1], x_2= roots[2,2])==0],x_3);
roots = roots.subs(aux[0]);
```

```
roots[(2),(0,1,2,3)] = \
roots[(2),(0,1,2,3)].substitute(rho = rho_minus , u_1 = u_1_minus, p_rho = p_rho_minus, xi=1);

#-----------------------------------------
#x_0 = roots[3,0] =  -1
#-----------------------------------------

roots[3,1]= x_1;
aux = solve([Q_1(x_0 = roots[3,0])==0],x_1);
roots = roots.subs(aux[0]);

roots[3,2]= x_2;
aux = solve([Q_2(x_0= roots[3,0],x_1=roots[3,1])==0],x_2);
roots = roots.subs(aux[0]);
roots[3,2] = \
roots[3,2].substitute( -1/2*(lambda_0^4*rho - 4*lambda_0^3*rho*u_1 + 6*lambda_0^2*rho*u_1^2 \
                          - 4*lambda_0*rho*u_1^3 + rho*u_1^4)\
                      /(lambda_0^2 - 2*lambda_0*u_1 + u_1^2 - p_rho) == -rho*(lambda_0 -u_1)^4/(2*((lambda_0 -u_1)^2 - p_
rho))); ##OK

roots[3,3]= x_3;
aux = solve([Q_3(x_0= roots[3,0],x_1=roots[3,1], x_2= roots[3,2])==0],x_3);
roots = roots.subs(aux[0]);

roots[(3),(0,1,2,3)] = roots[(3),(0,1,2,3)].substitute(rho = rho_plus , u_1 = u_1_plus, p_rho = p_rho_plus, xi=1);
```

Decaying manifolds and the spectrum of $S^{\pm}$

By [ZS], we know that the basis for the decaying manifolds are given by solving the following ode:

$$X_{x_1}(x_1,\epsilon) = -A_1^{(-1)}(\lambda A_0 + i\xi A_2)^{\pm} X(x_1,\epsilon) \quad , \quad \text{for } x_1 \gtrless 0.$$

On the previous section we computed the spectrum of $-A_1^{(-1)}(\lambda A_0 + i\xi A_2)^{\pm}$. Now, we can summarize these results.

> Observation 10: In what follows, we will choose $\alpha = \bar{u}_1^+$.

Now we can spectify things a little bit, since we need to analyse the signal of the eigenvalues in the positive and negative side.

$$\sigma(S^+) = \left\{ \underbrace{\pm 1 + \mathcal{O}(\epsilon^2)}_{\gtrless 0}, \underbrace{-\frac{1}{u_1^+ + \sqrt{p_{\rho^+}}}(\lambda_0 + \epsilon\lambda_1) + \mathcal{O}(\epsilon^2)}_{<0}, \underbrace{-\frac{1}{u_1^+ - \sqrt{p_{\rho^+}}}(\lambda_0 + \epsilon\lambda_1) + \mathcal{O}(\epsilon^2)}_{>0}, \underbrace{-\frac{\lambda}{\alpha}}_{<0} \right\}$$

$$\sigma(S^-) = \left\{ \underbrace{\pm 1 + \mathcal{O}(\epsilon^2)}_{\lessgtr}, \underbrace{-\frac{1}{u_1^- + \sqrt{p_{\rho^-}}}(\lambda_0 + \epsilon\lambda_1) + \mathcal{O}(\epsilon^2)}_{<0}, \underbrace{-\frac{1}{u_1^- - \sqrt{p_{\rho^-}}}(\lambda_0 + \epsilon\lambda_1) + \mathcal{O}(\epsilon^2)}_{<0}, \underbrace{-\frac{\lambda}{\alpha}}_{<0} \right\}$$

As we are looking for \textit{decaying manifolds} we must have, for $x_1 > 0$, eigenspaces associated to the following eigenvalues:

$$\{\mu_1,\mu_2,\mu_3\} = \left\{ -1 + \mathcal{O}(\epsilon^2), -\frac{1}{u_1^+ + \sqrt{p_{\rho^+}}}(\lambda_0 + \epsilon\lambda_1) + \mathcal{O}(\epsilon^2), \underbrace{-\frac{\lambda}{\alpha}}_{<0} \right\}$$

Analogously, \textit{decaying} manifolds for $x_1 < 0$ must be eigenspaces associated to the following eigenvalues:

$$\mu_4 = \left\{ +1 + \mathcal{O}(\epsilon^2) \right\}$$

> **Remark 10:** Notice that the number of eigenvalues in each interior $x_1 \gtrless 0$ is consistent with the analysis derived from Hersh's lemma [2]

This part is carried out in the following piece of code

In [7]:
```
var('a alpha h_1 lambd lambda_0 lambda_1 lambda_2 lambda_3 x_0 x_1 x_2 x_3 x  mu gamm')
### The eigenvector x = x(mu) as function of the eigenvalue mu is  given by

##x_1>0   RIGHT SIDE

x_mu_plus = x_mu.substitute(rho = rho_plus, u_1= u_1_plus, p_rho=  p_rho_plus, xi=1) ;
x_plus=  x_mu_plus.substitute(mu=x_0+epsilon*x_1+epsilon^2*x_2 + epsilon^3*x_3,lambd= lambda_0 + epsilon*lambda_1+ epsilo
n^2*lambda_2 + epsilon^3*lambda_3);
x_plus = expand(x_plus);

#-----------------------------------------------------------------------------------
#Defining an auxiliary function that collects entries
#-----------------------------------------------------------------------------------
def get_coef(v,x,p,entry):
    #out = (v[entry][0]).coeff(x,p)     ##... after a few hours trying to figure out how to deal with indexing
    out = (v[entry][0]).coefficient(x,p)     ##... after a few hours trying to figure out how to deal with indexing
    return out

#-----------------------------------------------------------------------------------

y_plus_zero = matrix([[get_coef(x_plus,epsilon,0,0)], [get_coef(x_plus,epsilon,0,1)], \
                      [get_coef(x_plus,epsilon,0,2)], [get_coef(x_plus,epsilon,0,3)], \
                      [get_coef(x_plus,epsilon,0,4)]]);
y_plus_one = matrix([[get_coef(x_plus,epsilon,1,0)], [get_coef(x_plus,epsilon,1,1)],\
                     [get_coef(x_plus,epsilon,1,2)], [get_coef(x_plus,epsilon,1,3)], \
                     [get_coef(x_plus,epsilon,1,4)]]);
y_plus_two = matrix([[get_coef(x_plus,epsilon,2,0)], [get_coef(x_plus,epsilon,2,1)], \
                     [get_coef(x_plus,epsilon,2,2)], [get_coef(x_plus,epsilon,2,3)],\
                     [get_coef(x_plus,epsilon,2,4)]]);
y_plus_three = matrix([[get_coef(x_plus,epsilon,3,0)], [get_coef(x_plus,epsilon,3,1)], \
                       [get_coef(x_plus,epsilon,3,2)], [get_coef(x_plus,epsilon,3,3)], \
                       [get_coef(x_plus,epsilon,3,4)]]);
```

That is, the eigenvector X_plus admits an expansion of the type

$$X(\mu)^{+}X\_plus = y\_plus\_zero + y\_plus\_one + y\_plus\_two\epsilon^2 + y\_plus\_three\epsilon^3 + O(\epsilon^4),$$

with a similar expression for the negative eigenvector $X(\mu)^{-}$.

For instance, we can see below the result for the 0th order term in this expansion

In [8]:
```
y_plus_one
```

Out[8]:
```
[3*I*u_1_plus^2*x_0^2*x_1/p_rho_plus + I*lambda_1*u_1_plus*x_0^2/p_rho_plus + 2*I*lambda_0*u_1_plus*x_0*x_1/p_rho_
plus - I*u_1_plus^2*x_1/p_rho_plus - I*lambda_1*u_1_plus/p_rho_plus]
[                                                                                                          I*lambda_1*x
_0^2/rho_plus + 2*I*lambda_0*x_0*x_1/rho_plus - I*lambda_1/rho_plus]
[
           -u_1_plus^2*x_0^2 - 2*lambda_0*u_1_plus*x_0 - lambda_0^2]
[
                               I*u_1_plus^2*x_0 + I*lambda_0*u_1_plus]
[
                                   lambda_0*u_1_plus*x_0 + lambda_0^2]
```

The Lopatinski determinant

A quick look at conjugated systems of ode's.

We know that, if we are solving the problem

$$\dot{X} = AX$$

then the solutions to the system

$$\dot{Y} = BY,$$

where $B = PAP^{-1}$ are homeomorphic to the solutions X; furthermore, $PX = Y$. In our present context, since we did some diagonalizations, we must perform some changes in the $\lambda[f_0] + i\xi[x_2]$ column. In our particular case, we used a matrix $P = DP_2P_1$, i.e.,

$$P = DP_2P_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{p_\rho}{\rho u_1} & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{h_1}{u_1} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{h_1}{u_1} & 0 & 1 \end{pmatrix},$$

which has a nonvanishing determinant. It turns out then that

$$\begin{aligned} \Delta(\lambda, \xi) &= det(A_1^+ P_+^{-1} x(\mu_1), A_1^+ P_+^{-1} x(\mu_2), A_1^+ P_+^{-1} x(\mu_3), A_1^- P_-^{-1} x(\mu_4), \lambda[f_0] + i\xi[f_2])\Big|_{x_1=0} \\ &= det\left(A_1^+ P_+^{-1}\right) det(x(\mu_1), x(\mu_2), x(\mu_3), P_+(A_1^+)^{-1} A_1^- P_-^{-1} x(\mu_4), P_+(A_1^+)^{-1}(\lambda[f_0] + i\xi[f_2])\Big)\Big|_{x_1=0} = \\ &= \Delta_0 + \epsilon\Delta_1 + \epsilon^2\Delta_2 + \dots \end{aligned} \tag{10}$$

The above formula turns out to be more convenient for our calculations; that is the formula we adopted here.

The Lopatisnki Determinant and its ϵ expansion

The Lopatinski determinant is given by the following formula:

$$\Delta(\lambda, \xi) = det(\mathcal{R}^+, \mathcal{R}^-, \lambda[f_0] + i\xi[f_2])\Big|_{x_1=0} \tag{1}$$

where $\mathcal{R}^\pm$ are basis for the x_1 $><= 0$ manifolds, given on the previous section. Our job in this section is to find out if $\Delta(\lambda, \xi) = 0$ fora $\lambda > 0$ as $h_1 \to \infty$ (i.e., $\epsilon \to 0$). The analysis then reduces to find $\lambda > 0$ so that

$$0 = \tilde{\Delta_0} + \epsilon\tilde{\Delta_1} + \epsilon^2\tilde{\Delta_2} + \dots \tag{11}$$

> Remark 11 (multilinearity): In what follows we strongly use the multilinearity of the determinant function. It is not hard to find out that if X_i denotes the i^{th} column of the the matrix $A = [X_1, X_2, \dots X_n]$ and $X_i(\epsilon) = X_i^{(0)} + \epsilon X_i^{(1)} + \epsilon^2(X_i^{(2)}) + \dots$ then
>
> $$det(A) = det(X_1^{(0)}, \dots, X_n^{(0)}) + \epsilon A_1 + \epsilon^2 A_2 \dots$$
>
> where $A_k = \sum_{a_1+\dots+a_n=k} det(X_1^{(a_1)}, \dots, X_n^{(a_n)})$. Therefore, in which concerns to equation (11) we must analize
>
> $$\Delta_n = 0,$$
>
> for all $n \in \mathbb{N}$, where
>
> $$\Delta_n = \sum_{a_1+\dots+a_5=n} det(X_1^{(a_1)}, X_2^{(a_2)}, X_3^{(a_3)}, X_4^{(a_4)}, X_5^{(a_5)}). \tag{12}$$

From the previous Remark we conclude that $\Delta_0 = \Delta_1 = 0$.

In [9]:
```
#########################################################################
### Calculating and approximated basis for the decaying manifolds
#########################################################################

#-----------------------------------------
# basis for x_1 > 0
#-----------------------------------------

X_1_zero = y_plus_zero.substitute(x_0 = roots[3,0], x_1 = roots[3,1] , x_2 = roots[3,2]);
X_1_one  =  y_plus_one.substitute(x_0 = roots[3,0], x_1 = roots[3,1] , x_2 = roots[3,2]);
X_1_two  =  y_plus_two.substitute(x_0 = roots[3,0], x_1 = roots[3,1] , x_2 = roots[3,2]);

#-------------------------------------------------------------------------------

X_2_zero = y_plus_zero.substitute(x_0 = roots[0,0], x_1 = roots[0,1] , x_2 = roots[0,2]);
X_2_one  =  y_plus_one.substitute(x_0 = roots[0,0], x_1 = roots[0,1] , x_2 = roots[0,2]);
X_2_two  =  y_plus_two.substitute(x_0 = roots[0,0], x_1 = roots[0,1] , x_2 = roots[0,2]);

#-------------------------------------------------------------------------------------
## the eigenvector associated to the eigenvalue lambda/u_1_plus

X_3     =  matrix([[0],[0],[0],[-lambd],[i*u_1_plus ]])
X_3     =  X_3(lambd= lambda_0 + epsilon*lambda_1+ epsilon^2*lambda_2);

X_3_zero = matrix( [[0],[ 0],[ 0],[ -lambda_0],[I*u_1_plus]]);
X_3_one  = matrix( [[0],[ 0],[ 0],[ -lambda_1],[0]]);
X_3_two  = matrix( [[0],[ 0],[ 0],[ -lambda_2],[0]]);
```

For example, the third column is

In [10]:
```
X_3
```

Out[10]:
```
[                                                0]
[                                                0]
[                                                0]
[-epsilon^2*lambda_2 - epsilon*lambda_1 - lambda_0]
[                                       I*u_1_plus]
```

which corresponds, if we expand in terms of ϵ will give us

$$X_3 = X\_3\_zero + X\_3\_one * \epsilon + X\_3\_two * \epsilon^2$$

where

In [11]:
```
X_3_zero
```

Out[11]:
```
[         0]
[         0]
[         0]
[ -lambda_0]
[I*u_1_plus]
```

In [12]:
```
X_3_one
```

Out[12]:
```
[        0]
[        0]
[        0]
[-lambda_1]
[        0]
```

In [13]:
```
X_3_two
```

Out[13]:
```
[        0]
[        0]
[        0]
[-lambda_2]
[        0]
```

Now we evaluate the 4th column:

In [14]:
```
### Change of basis (explained in the section about conjugated systems in the pdf's appendix)

##x_1>0

P_plus = D*P_2*P_1 ;
P_plus = \
P_plus.substitute(rho = rho_plus, u_1= u_1_plus, p_rho=  p_rho_plus, xi=1, h_1 =1/epsilon) ;
A_1_plus_inverse = \
A_1_inv.substitute(rho = rho_plus, u_1= u_1_plus, p_rho=  p_rho_plus, xi=1, h_1 =1/epsilon) ;

##x_1 < 0

P_minus = D*P_2*P_1 ;
P_minus = P_minus.substitute(rho = rho_minus, u_1= u_1_minus, p_rho=  p_rho_minus, xi=1, h_1= 1/epsilon);
A_1_minus = A_1.substitute(rho = rho_minus, u_1= u_1_minus, p_rho=  p_rho_minus, xi=1, h_1= 1/epsilon);

### The matrices we are going to use are the following

Aux_matrix_Lop_plus = P_plus*A_1_plus_inverse;
Aux_matrix_Lop_plus = \
Aux_matrix_Lop_plus.substitute(-(u_1_plus^2 + p_rho_plus)/((u_1_plus^2 - p_rho_plus)*rho_plus) + 2*p_rho_plus\
                               /((u_1_plus^2 - p_rho_plus)*rho_plus) == -1/rho_plus);  ##OK
Aux_matrix_Lop_plus = \
Aux_matrix_Lop_plus.substitute(u_1_plus/((u_1_plus^2 - p_rho_plus)*rho_plus) - p_rho_plus\
                               /((u_1_plus^2 - p_rho_plus)*rho_plus*u_1_plus) == 1/(rho_plus*u_1_plus));   ###OK
Aux_matrix_Lop_plus = \
Aux_matrix_Lop_plus.substitute(rho_plus*u_1_plus\
                               /(rho_plus*u_1_plus^2 - 1/epsilon^2) - \
                               1/((rho_plus*u_1_plus^2 - 1/epsilon^2)*epsilon^2*u_1_plus) == 1/u_1_plus);   ##OK

Aux_matrix_Lop_minus = A_1_minus*P_minus.inverse();
```

It is worth to not lose track of why we are doing all these simplifications. Indeed, an evaluation of the matrix P_1^- gives

In [15]:
```
P_minus
```

Out[15]:
```
[                                1                                0                                0
                    0                                0]
[p_rho_minus/(rho_minus*u_1_minus)                                1                                0
                    0                                0]
[                                0                                0             1/(epsilon*u_1_minus)
                    0                                0]
[                                0                                0                                0
                    1                                0]
[                                0                                0            -1/(epsilon*u_1_minus)
                    0                                1]
```

In [16]:
```
#-----------------------------------------
# basis for x_1 < 0
#-----------------------------------------

Aux_column_4 = Aux_matrix_Lop_plus*Aux_matrix_Lop_minus;
#Aux_column_4(x_2 = roots[2,2],x_1 = roots[2,1],x_0 = roots[2,0])

### Substitutions withouth expansions
Aux_column_4 = Aux_column_4.substitute(2*(u_1_minus - p_rho_minus/u_1_minus)*u_1_plus/(u_1_plus^2 - p_rho_plus) - (u_1_mi
nus^2 - p_rho_minus)/(u_1_plus^2 - p_rho_plus) == (u_1_minus^2 - p_rho_minus)*(2*u_1_plus/u_1_minus -1)/(u_1_plus^2 - p_r
ho_plus));    ##OK
Aux_column_4 = Aux_column_4.substitute(-2*rho_minus*u_1_minus/(u_1_plus^2 - p_rho_plus) + 2*rho_minus*u_1_plus/(u_1_plus^
2 - p_rho_plus) == 2*rho_minus*(u_1_plus -u_1_minus)/(u_1_plus^2 - p_rho_plus)); ##OK
Aux_column_4 = Aux_column_4.substitute(-rho_minus/rho_plus + 2*rho_minus*u_1_minus/(rho_plus*u_1_plus) == rho_minus/rho_p
lus*(2*u_1_minus/u_1_plus -1));   ##OK
Aux_column_4 = Aux_column_4.substitute( -(u_1_minus - p_rho_minus/u_1_minus)/rho_plus + (u_1_minus^2 - p_rho_minus)/(rho_
plus*u_1_plus)  ==  (u_1_minus^2 - p_rho_minus)*(1/(rho_plus*u_1_plus) - 1/(rho_plus*u_1_minus)) );
Aux_column_4 = Aux_column_4.substitute(-1/((rho_plus*u_1_plus^2 - 1/epsilon^2)*epsilon^2) + u_1_minus/((rho_plus*u_1_plus
^2 - 1/epsilon^2)*epsilon^2*u_1_plus) ==        1/(1 - epsilon^2*rho_plus*u_1_plus^2 )*(1 -  u_1_minus/u_1_plus));
Aux_column_4 = Aux_column_4.substitute((epsilon*rho_minus*u_1_minus^2 - 1/epsilon)/((rho_plus*u_1_plus^2 - 1/epsilon^2)*e
psilon) ==
                                       (1 - epsilon^2*rho_minus*u_1_minus^2 )/(1 - epsilon^2*rho_plus*u_1_plus^2 ));

### simplification related to expansions
Aux_column_4 = Aux_column_4.substitute((1 - epsilon^2*rho_minus*u_1_minus^2 )/(1 - epsilon^2*rho_plus*u_1_plus^2 ) == 1 +
 epsilon^2*(rho_plus*u_1_plus^2 - rho_minus*u_1_minus^2));   ##OK
Aux_column_4 = Aux_column_4.substitute( 1/(1 - epsilon^2*rho_plus*u_1_plus^2 )*(1 -  u_1_minus/u_1_plus) ==(1 + epsilon^2
*rho_plus*u_1_plus^2 )*(1 -  u_1_minus/u_1_plus))   ##OK
```

In [17]:
```
x_mu_minus = x_mu.substitute(rho = rho_minus, u_1= u_1_minus, p_rho=  p_rho_minus, xi=1);
x_minus=  x_mu_minus\
.substitute(mu=x_0+epsilon*x_1+epsilon^2*x_2,lambd= lambda_0 + epsilon*lambda_1+ epsilon^2*lambda_2);

x_minus = Aux_column_4*x_minus;

x_minus = expand(x_minus);
```

Now we expand

In [18]:
```
y_minus_zero = matrix([[get_coef(x_minus,epsilon,0,0)], [get_coef(x_minus,epsilon,0,1)], \
        [get_coef(x_minus,epsilon,0,2)], [get_coef(x_minus,epsilon,0,3)], [get_coef(x_minus,epsilon,0,4)]]);

y_minus_one = matrix([[get_coef(x_minus,epsilon,1,0)], [get_coef(x_minus,epsilon,1,1)], \
                      [get_coef(x_minus,epsilon,1,2)], [get_coef(x_minus,epsilon,1,3)], \
                      [get_coef(x_minus,epsilon,1,4)]]);

y_minus_two = matrix([[get_coef(x_minus,epsilon,2,0)], [get_coef(x_minus,epsilon,2,1)], \
                      [get_coef(x_minus,epsilon,2,2)], [get_coef(x_minus,epsilon,2,3)], \
                      [get_coef(x_minus,epsilon,2,4)]]);

y_minus_three = matrix([[get_coef(x_minus,epsilon,3,0)], [get_coef(x_minus,epsilon,3,1)],\
                        [get_coef(x_minus,epsilon,3,2)], [get_coef(x_minus,epsilon,3,3)],\
                        [get_coef(x_minus,epsilon,3,4)]]);
```

which gives, if we evaluate, things like

In [19]:
```
y_minus_zero
```

Out[19]:
```
[-I*u_1_minus^4*x_0^3/((u_1_plus^2 - p_rho_plus)*p_rho_minus) + 2*I*u_1_minus^3*u_1_plus*x_0^3/((u_1_plus^2 - p_rh
o_plus)*p_rho_minus) - I*lambda_0*u_1_minus^3*x_0^2/((u_1_plus^2 - p_rho_plus)*p_rho_minus) + 2*I*lambda_0*u_1_min
us^2*u_1_plus*x_0^2/((u_1_plus^2 - p_rho_plus)*p_rho_minus) + I*u_1_minus^2*x_0^3/(u_1_plus^2 - p_rho_plus) - 2*I*
u_1_minus*u_1_plus*x_0^3/(u_1_plus^2 - p_rho_plus) + I*u_1_minus^4*x_0/((u_1_plus^2 - p_rho_plus)*p_rho_minus) - 2
*I*u_1_minus^3*u_1_plus*x_0/((u_1_plus^2 - p_rho_plus)*p_rho_minus) - I*lambda_0*u_1_minus*x_0^2/(u_1_plus^2 - p_r
ho_plus) + I*lambda_0*u_1_minus^3/((u_1_plus^2 - p_rho_plus)*p_rho_minus) - 2*I*lambda_0*u_1_minus^2*u_1_plus/((u_
1_plus^2 - p_rho_plus)*p_rho_minus) - I*u_1_minus^2*x_0/(u_1_plus^2 - p_rho_plus) + 2*I*u_1_minus*u_1_plus*x_0/(u_
1_plus^2 - p_rho_plus) + I*lambda_0*u_1_minus/(u_1_plus^2 - p_rho_plus)]
[
                                                                                                                -I*
u_1_minus^3*x_0^3/(p_rho_minus*rho_plus) + I*u_1_minus^4*x_0^3/(p_rho_minus*rho_plus*u_1_plus) - I*lambda_0*u_1_mi
nus^2*x_0^2/(p_rho_minus*rho_plus) + I*lambda_0*u_1_minus^3*x_0^2/(p_rho_minus*rho_plus*u_1_plus) + I*u_1_minus*x_
0^3/rho_plus - I*u_1_minus^2*x_0^3/(rho_plus*u_1_plus) + I*u_1_minus^3*x_0/(p_rho_minus*rho_plus) - I*u_1_minus^4*
x_0/(p_rho_minus*rho_plus*u_1_plus) + I*lambda_0*u_1_minus*x_0^2/(rho_plus*u_1_plus) + I*lambda_0*u_1_minus^2/(p_r
ho_minus*rho_plus) - I*lambda_0*u_1_minus^3/(p_rho_minus*rho_plus*u_1_plus) - I*u_1_minus*x_0/rho_plus + I*u_1_min
us^2*x_0/(rho_plus*u_1_plus) - I*lambda_0*u_1_minus/(rho_plus*u_1_plus)]
[
                                                                                                                0]
[
                                                                                                                0]
[
                                                                                                                0]
```

which, definitely, is not pleasant to the eyes. Now we substitute the expansion for the eigenvalue, which corresponds to

$$\mu = x\_0 + x\_1 * \epsilon + x\_2 * \epsilon^2 + O(\epsilon^3)$$

where $x_0 = roots[2,0], x_1 = roots[2,1], x_2 = roots[2,2]$. Hence, we expand and collect lower order terms:

In [20]:
```
X_4_zero = y_minus_zero.substitute(x_0 = roots[2,0], x_1 = roots[2,1] , x_2 = roots[2,2]);
X_4_one  =  y_minus_one.substitute(x_0 = roots[2,0], x_1 = roots[2,1] , x_2 = roots[2,2]);
X_4_two  =  y_minus_two.substitute(x_0 = roots[2,0], x_1 = roots[2,1] , x_2 = roots[2,2]);
```

Unfortunately the result is really "ugly", so we need to intervene in order to simplify some of the expressions. That's what we do next:

In [21]:
```
##############################
### Attempt to simplify X_4_two
##############################

## FIRST COORDINATE

a=X_4_two[0,0];

### THE IDEA IS TO TREAT NUMERATOR AND DENOMINATOR SEPARATEDLY

num = numerator(X_4_two[0,0]);  ### check degree of these polynomials

num =   num.collect(u_1_plus);    ## degree 1
a_0 = num.coefficient(u_1_plus,0);
a_1 =num.coefficient(u_1_plus,1);           ## Therefore num = a_0 + a_1*u_1_plus


a_0 = a_0.collect(p_rho_minus);   ## degree 1
a_0_0 = a_0.coefficient(p_rho_minus,0);
a_0_0 = a_0_0.substitute( -I*lambda_0^3*rho_minus*u_1_minus^3\
                          - 3*I*lambda_0^2*rho_minus*u_1_minus^4 - 3*I*lambda_0*rho_minus*u_1_minus^5 \
                          - I*rho_minus*u_1_minus^6 == -i*u_1_minus^3*rho_minus*(lambda_0 + u_1_minus)^3); ##OK

a_0_1 = a_0.coefficient(p_rho_minus,1);     ##  Therefore a_0 = a_0_0 + a_0_1*p_rho_minus;
a_0_1 = a_0_1.substitute(-I*lambda_0^3*rho_minus*u_1_minus - I*lambda_0^2*rho_minus*u_1_minus^2 \
                          + I*lambda_0*rho_minus*u_1_minus^3 + I*rho_minus*u_1_minus^4 \
                          == -i*rho_minus*u_1_minus*(lambda_0 - u_1_minus)*(lambda_0 + u_1_minus)^2);  ##OK

a_0 = a_0_0 + a_0_1*p_rho_minus;

a_0 = -I*rho_minus*u_1_minus*(lambda_0 +u_1_minus)^2*(u_1_minus^2*(lambda_0+u_1_minus)\
                                                      +(lambda_0 -u_1_minus)*p_rho_minus);

#a_0 = a_0.substitute(-I*(lambda_0 + u_1_minus)^3*rho_minus*u_1_minus^3 - I*(lambda_0 - u_1_minus)*(lambda_0 + u_1_minus)
^2*p_rho_minus*rho_minus*u_1_minus == -I*rho_minus*u_1_minus*(lambda_0 +u_1_minus)^2*(u_1_minus^2*(lambda_0+u_1_minus) +(
lambda_0 -u_1_minus)*p_rho_minus));  ## IT DIDN'T WORK!!!

a_1 = a_1.collect(p_rho_minus);
a_1_0 = a_1.coefficient(p_rho_minus,0);
a_1_0 = a_1_0.substitute(2*I*lambda_0^3*rho_minus*u_1_minus^2 + 6*I*lambda_0^2*rho_minus*u_1_minus^3\
                          + 6*I*lambda_0*rho_minus*u_1_minus^4 + 2*I*rho_minus*u_1_minus^5 \
                          == i*2*rho_minus*u_1_minus^2*(lambda_0 + u_1_minus)^3);   ##OK

a_1_1 = a_1.coefficient(p_rho_minus,1);     ## Therefore a_1 = a_1_0 + a_1_1*p_rho_minus;
a_1_1 = a_1_1.substitute(-2*I*lambda_0^2*rho_minus*u_1_minus - 4*I*lambda_0*rho_minus*u_1_minus^2\
                          - 2*I*rho_minus*u_1_minus^3 == -i*2*rho_minus*u_1_minus*(lambda_0 + u_1_minus)^2);   ##OK

a_1 = a_1_0 + a_1_1*p_rho_minus;

a_1= a_1.substitute(2*I*(lambda_0 + u_1_minus)^3*rho_minus*u_1_minus^2 \
                    - 2*I*(lambda_0 + u_1_minus)^2*p_rho_minus*rho_minus*u_1_minus ==  \
                    2*i*(lambda_0 + u_1_minus)^2*rho_minus*u_1_minus*(u_1_minus*(lambda_0 +u_1_minus)\
                                                                      -p_rho_minus));  ##OK

num = a_0 + a_1*u_1_plus;
num = num.substitute(2*I*(lambda_0 + u_1_minus)^2*((lambda_0 + u_1_minus)*u_1_minus - p_rho_minus)*\
                     rho_minus*u_1_minus*u_1_plus - I*(lambda_0 + u_1_minus)^2*\
                     ((lambda_0 + u_1_minus)*u_1_minus^2 + (lambda_0 - u_1_minus)*\
                      p_rho_minus)*rho_minus*u_1_minus == i*(lambda_0 + u_1_minus )^2*\
                     rho_minus*u_1_minus*(2*((lambda_0 + u_1_minus)*u_1_minus - p_rho_minus)*\
                                          u_1_plus - ((lambda_0 + u_1_minus)*u_1_minus^2 + \
                                                      (lambda_0 - u_1_minus)*p_rho_minus)));


den = denominator(X_4_two[0,0]);
den = den.substitute((lambda_0^2 + 2*lambda_0*u_1_minus + u_1_minus^2 - p_rho_minus)\
                     == ( (lambda_0 + u_1_minus)^2 - p_rho_minus));  ##OK

#w = X_4_two[0,0];
#w = -w + num/den;
#w.substitute(u_1_minus=.2, u_1_plus = 2, rho_plus=3, rho_minus=.5, p_rho_plus=7, p_rho_minus=3, lambda_0=1, lambda_1=4,
lambda_2=.3)

X_4_two[0,0] = num/den;    #OK

#### Attempt to simplify X_4_two

## SECOND COORDINATE

a=X_4_two[1,0];
num = numerator(X_4_two[1,0]);
num =   num.collect(u_1_plus);
```

```
a_0 = num.coefficient(u_1_plus,0);
a_1 =num.coefficient(u_1_plus,1);           ## Therefore num = a_0 + a_1*u_1_plus

a_0 = a_0.collect(p_rho_minus);
a_0_0 = a_0.coefficient(p_rho_minus,0);
a_0_0= a_0_0.substitute(I*lambda_0^3*rho_minus*u_1_minus^3 \
                        + 3*I*lambda_0^2*rho_minus*u_1_minus^4 + 3*I*lambda_0*rho_minus*u_1_minus^5 \
                        + I*rho_minus*u_1_minus^6 == i*rho_minus*u_1_minus^3*(lambda_0 +u_1_minus)^3); ##OK



a_0_1 = a_0.coefficient(p_rho_minus,1);     ##  Therefore a_0 = a_0_0 + a_0_1*p_rho_minus;
a_0_1 =a_0_1.substitute(I*lambda_0^3*rho_minus*u_1_minus + I*lambda_0^2*rho_minus*u_1_minus^2 \
                        - I*lambda_0*rho_minus*u_1_minus^3 - I*rho_minus*u_1_minus^4  \
                        == i*rho_minus*u_1_minus*(lambda_0 - u_1_minus)*(lambda_0 +u_1_minus)^2);   ##OK

a_0 = a_0_0 + a_0_1*p_rho_minus;
a_0 =  I*(lambda_0 + u_1_minus)^2*rho_minus*u_1_minus*( (lambda_0 - u_1_minus)*p_rho_minus \
                                                      + (lambda_0 + u_1_minus)*u_1_minus^2 );


a_1 = a_1.collect(p_rho_minus);
a_1_0 = a_1.coefficient(p_rho_minus,0);
a_1_0=a_1_0.substitute(-I*lambda_0^3*rho_minus*u_1_minus^2 - 3*I*lambda_0^2*rho_minus*u_1_minus^3 \
                       - 3*I*lambda_0*rho_minus*u_1_minus^4 - I*rho_minus*u_1_minus^5 \
                       ==  - i*rho_minus*u_1_minus^2*(lambda_0 +u_1_minus)^3);    ##OK

a_1_1 = a_1.coefficient(p_rho_minus,1);     ##  Therefore a_0 = a_0_0 + a_0_1*p_rho_minus;
a_1_1 =a_1_1.substitute(I*lambda_0^2*rho_minus*u_1_minus + 2*I*lambda_0*rho_minus*u_1_minus^2 \
                        + I*rho_minus*u_1_minus^3 == i*rho_minus*u_1_minus*(lambda_0 + u_1_minus)^2); ##OK

a_1 = a_1_0 + a_1_1*p_rho_minus;

a_1 = a_1.substitute(-I*(lambda_0 + u_1_minus)^3*rho_minus*u_1_minus^2 + I*(lambda_0 + u_1_minus)^2\
                     *p_rho_minus*rho_minus*u_1_minus ==  I*(lambda_0 + u_1_minus)^2*rho_minus*u_1_minus\
                     *(p_rho_minus - u_1_minus*(lambda_0 + u_1_minus)) );    ##OK

num = a_0 + a_1*u_1_plus;
num = num.substitute(-I*(lambda_0 + u_1_minus)^2*((lambda_0 + u_1_minus)*u_1_minus \
                                                - p_rho_minus)*rho_minus*u_1_minus*u_1_plus \
                     + I*(lambda_0 + u_1_minus)^2*((lambda_0 + u_1_minus)*u_1_minus^2 \
                                                 + (lambda_0 - u_1_minus)*p_rho_minus)*rho_minus*u_1_minus \
                     == I*(lambda_0 + u_1_minus)^2*rho_minus*u_1_minus*(- ((lambda_0 + u_1_minus)*u_1_minus\
                                                                          - p_rho_minus)*u_1_plus \
                                                                       +  ((lambda_0 + u_1_minus)*u_1_minus^2 \
                                                                           + (lambda_0 - u_1_minus)*p_rho_minus)));

den = denominator(X_4_two[1,0]);
den = den.substitute((lambda_0^2 + 2*lambda_0*u_1_minus + u_1_minus^2 - p_rho_minus)*rho_plus*u_1_plus\
                     == ( (lambda_0 + u_1_minus)^2 - p_rho_minus)*rho_plus*u_1_plus);   ##OK

#w = X_4_two[1,0];
#w = -w + num/den;
#w.substitute(u_1_minus=.2, u_1_plus = 2, rho_plus=3, rho_minus=.5, p_rho_plus=7, p_rho_minus=3, lambda_0=1, lambda_1=4,
lambda_2=.3)

X_4_two[1,0] = num/den;   #OK

#w = X_4_two[1,0];
#w = -w + a;
#w.substitute(u_1_minus=.2, u_1_plus = 2, rho_plus=3, rho_minus=.5, p_rho_plus=7, p_rho_minus=3, lambda_0=1, lambda_1=4,
lambda_2=.3)
##w =a;
#w = -w +  X_4_two[1,0];
#w.substitute(u_1_minus =.2, p_rho_minus =3, rho_plus =2, u_1_plus=3,rho_minus=.33)
```

And finally, we evaluate the last column.

In [22]:
```
#-----------------------------------------------------------------------
# The column that comes from the Rankine Hugoniot condition is
#-----------------------------------------------------------------------

Aux_matrix_Lop_plus = Aux_matrix_Lop_plus.substitute(1/((rho_plus*u_1_plus^2 - 1/epsilon^2)*epsilon) \
                                                     == -epsilon*(1 + epsilon^2*rho_plus*u_1_plus^2));  ##OK
Aux_matrix_Lop_plus = Aux_matrix_Lop_plus.substitute(1/((rho_plus*u_1_plus^2 - 1/epsilon^2)*epsilon^2*u_1_plus)\
                                                     == -(1 + epsilon^2*rho_plus*u_1_plus^2)/u_1_plus); ##OK
X_5 = matrix([[lambd*(rho_plus - rho_minus)],[0], [i*(p_rho_plus*rho_plus - p_rho_minus*rho_minus)/gamm ],\
              [0], [0]])
X_5 = Aux_matrix_Lop_plus*X_5;
X_5 = X_5(lambd= lambda_0 + epsilon*lambda_1 + epsilon^2*lambda_2);
X_5 = X_5.expand();

X_5_zero = matrix([[get_coef(X_5,epsilon,0,0)], [get_coef(X_5,epsilon,0,1)],\
                   [get_coef(X_5,epsilon,0,2)], [get_coef(X_5,epsilon,0,3)], [get_coef(X_5,epsilon,0,4)]]);
X_5_one  = matrix([[get_coef(X_5,epsilon,1,0)], [get_coef(X_5,epsilon,1,1)], \
                   [get_coef(X_5,epsilon,1,2)], [get_coef(X_5,epsilon,1,3)], [get_coef(X_5,epsilon,1,4)]]);
X_5_two  = matrix([[get_coef(X_5,epsilon,2,0)], [get_coef(X_5,epsilon,2,1)],\
                   [get_coef(X_5,epsilon,2,2)], [get_coef(X_5,epsilon,2,3)], [get_coef(X_5,epsilon,2,4)]]);
```

For instance, the terms above denote the expansion of the eigenvectors $X(\mu)^{\pm}$ in terms of ϵ

We now study each term in the expansion of the Lopatinski determinant.

Eigenvectors as a function of eigenvalues: the parallel case .

In this section we derive explicitly an expression for the eigenvectors as a function of the eigenvalues. Moreover, we compute the basis for the decaying manifolds. We shall parametrize the eigenvectors as a functions of the depending eigenvalue; for later use, we shall evaluate the cases in which $\lambda_0 = 0$ and $\lambda_1 = 0$.

In what follows, we write

$$X_k = X_k^{(0)} + \epsilon X_k^{(1)} + \epsilon^2 X_k^{(2)} + \dots$$

Initially, notice that we can expand the eigenvector $x(\mu)$ given in (8) in ϵ terms. In fact, taking $\mu = x_0 + \epsilon x_1 + \epsilon^2 x_2$ and $\lambda = \lambda_0 + \epsilon\lambda_1 + \epsilon^2\lambda_2 + \dots$ we have (up to second order terms)

$$x = x(x_0 + \epsilon x_1 + \epsilon^2 x_2 + \dots) =$$

In order to make our life simpler, the table below describes the vectors that are zero as the components of the expansion $\lambda = \lambda_0 + \epsilon\lambda_1 + \dots$ are zero

$\forall \lambda_0, \lambda_1$	$X_1^{(0)} = X_4^{(0)} = \textbf{0}$
$\lambda_0 = 0$	$X_1^{(0)} = X_2^{(0)} = X_4^{(0)} = X_5^{(0)} = \textbf{0}$
$\lambda_0 = \lambda_1 = 0$	$X_1^{(0)} = X_2^{(0)} = X_2^{(1)} = X_3^{(1)} = X_4^{(0)} = X_5^{(0)} = \textbf{0}$

Table 1: evaluation of the eigenvector X_μ in the case $\lambda_0 = 0$ and $\lambda_0 = \lambda_1 = 0$

Eigenvalue $\mu = -1 - \epsilon^2 \frac{\left(\lambda_0 - u_{1+}\right)^4 \rho^+}{2\left(\left(\lambda_0 - u_{1+}\right)^2 - p_{\rho^+}\right)}$

$$X_1^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad X_1^{(1)} = \begin{pmatrix} 0 \\ 0 \\ -\lambda_0^2 + 2\lambda_0 u_{1^+} - u_{1^+}^2 \\ i\lambda_0 u_{1^+} - i u_{1^+}^2 \\ \lambda_0^2 - \lambda_0 u_{1^+} \end{pmatrix}, \quad X_1^{(2)} = \begin{pmatrix} \frac{i\left(\lambda_0 - u_{1+}\right)^4 \lambda_0 \rho^+ u_{1+}}{\left(\left(\lambda_0 - u_{1+}\right)^2 - p_{\rho^+}\right) p_{\rho^+}} - \frac{i\left(\lambda_0 - u_{1+}\right)^4 \rho^+ u_{1+}^2}{\left(\left(\lambda_0 - u_{1+}\right)^2 - p_{\rho^+}\right) p_{\rho^+}} - \frac{i\lambda_0^3 \rho^+ u_{1+}}{p_{\rho^+}} + \frac{3i\lambda_0^2 \rho^+ u_{1+}^2}{p_{\rho^+}} - \frac{3i\lambda_0 \rho^+ u_{1+}^3}{p_{\rho^+}} + \frac{i\rho^+ u_{1+}^4}{p_{\rho^+}} \\ \frac{i\left(\lambda_0 - u_{1+}\right)^4 \lambda_0}{\left(\lambda_0 - u_{1+}\right)^2 - p_{\rho^+}} - i\lambda_0^3 + 2i\lambda_0^2 u_{1^+} - i\lambda_0 u_{1^+}^2 \\ -2\lambda_0\lambda_1 + 2\lambda_1 u_{1^+} \\ i\lambda_1 u_{1^+} \\ 2\lambda_0\lambda_1 - \lambda_1 u_{1^+} \end{pmatrix}$$

When $\lambda_0 = 0$ the vectors above become

$$X_1^{(0)}(\lambda_0 = 0) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad X_1^{(1)}(\lambda_0 = 0) = \begin{pmatrix} 0 \\ 0 \\ -u_{1^+}^2 \\ -i u_{1^+}^2 \\ 0 \end{pmatrix}, \quad X_1^{(2)}(\lambda_0 = 0) = \begin{pmatrix} -\frac{i\rho^+ u_{1+}^6}{\left(u_{1+}^2 - p_{\rho^+}\right) p_{\rho^+}} + \frac{i\rho^+ u_{1+}^4}{p_{\rho^+}} \\ 0 \\ 2\lambda_1 u_{1^+} \\ i\lambda_1 u_{1^+} \\ -\lambda_1 u_{1^+} \end{pmatrix}$$

When $\lambda_0 = \lambda_1 = 0$ the vectors above become

$$X_1^{(0)}(\lambda_0=\lambda_1=0)=\begin{pmatrix}0\\0\\0\\0\\0\end{pmatrix},\quad X_1^{(1)}(\lambda_0=\lambda_1=0)=\begin{pmatrix}0\\0\\-u_{1^+}^2\\-i\,u_{1^+}^2\\0\end{pmatrix},\quad X_1^{(2)}(\lambda_0=\lambda_1=0)=\begin{pmatrix}-\frac{i\,\rho^+u_{1^+}^6}{(u_{1^+}^2-p_{\rho^+})p_{\rho^+}}+\frac{i\,\rho^+u_{1^+}^4}{p_{\rho^+}}\\0\\0\\0\\0\end{pmatrix}$$

Eigenvalue $\mu=-\frac{\lambda_0+\lambda_1\epsilon}{u_1^+\sqrt{p_{\rho^+}}}+\mathcal{O}(\epsilon^2)$

$$X_2^{(0)}=\begin{pmatrix}\frac{i\,\lambda_0u_{1^+}^2}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)p_{\rho^+}}+\frac{i\,\lambda_0^3u_{1^+}}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)^2p_{\rho^+}}-\frac{i\,\lambda_0^3u_{1^+}^2}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)^3p_{\rho^+}}-\frac{i\,\lambda_0u_{1^+}}{p_{\rho^+}}\\-\frac{i\,\lambda_0^3}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)^2\rho^+}-\frac{i\,\lambda_0}{\rho^+}\\0\\0\\0\end{pmatrix},\quad X_2^{(1)}=\begin{pmatrix}\frac{i\,\lambda_1u_{1^+}^2}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)p_{\rho^+}}+\frac{3i\,\lambda_0^2\lambda_1u_{1^+}}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)^2p_{\rho^+}}-\frac{3i\,\lambda_0^2\lambda_1u_{1^+}^2}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)^3p_{\rho^+}}-\frac{i\,\lambda_1u_{1^+}}{p_{\rho^+}}\\-\frac{3i\,\lambda_0^2\lambda_1}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)^2\rho^+}-\frac{i\,\lambda_1}{\rho^+}\\\frac{2\,\lambda_0^2u_{1^+}}{u_{1^+}+\sqrt{p_{\rho^+}}}-\frac{\lambda_0^2u_{1^+}^2}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)^2}-\lambda_0^2\\-\frac{i\,\lambda_0u_{1^+}^2}{u_{1^+}+\sqrt{p_{\rho^+}}}+i\,\lambda_0u_{1^+}\\-\frac{\lambda_0^2u_{1^+}}{u_{1^+}+\sqrt{p_{\rho^+}}}+\lambda_0^2\end{pmatrix}$$

$X_2^{(2)}$ has a very long expression, so it will not be written here.

When $\lambda_0=0$ the vectors above become

$$X_2^{(0)}(\lambda_0=0)=\begin{pmatrix}0\\0\\0\\0\\0\end{pmatrix},\quad X_2^{(1)}(\lambda_0=0)=\begin{pmatrix}\frac{i\,\lambda_1u_{1^+}^2}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)p_{\rho^+}}-\frac{i\,\lambda_1u_{1^+}}{p_{\rho^+}}\\-\frac{i\,\lambda_1}{\rho^+}\\0\\0\\0\end{pmatrix},\quad X_2^{(2)}(\lambda_0=0)=\begin{pmatrix}\frac{i\,\lambda_2u_{1^+}^2}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)p_{\rho^+}}-\frac{i\,\lambda_2u_{1^+}}{p_{\rho^+}}\\-\frac{i\,\lambda_2}{\rho^+}\\0\\-\frac{i\,\lambda_1u_{1^+}^2}{u_{1^+}+\sqrt{p_{\rho^+}}}+i\,\lambda_1u_{1^+}\\0\end{pmatrix}$$

When $\lambda_0=\lambda_1=0$ the vectors above become

$$X_2^{(0)}(\lambda_0=\lambda_1=0)=X_2^{(1)}(\lambda_0=\lambda_1=0)=\begin{pmatrix}0\\0\\0\\0\\0\end{pmatrix},\quad X_2^{(2)}(\lambda_0=\lambda_1=0)=\begin{pmatrix}\frac{i\,\lambda_2u_{1^+}^2}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)p_{\rho^+}}-\frac{i\,\lambda_2u_{1^+}}{p_{\rho^+}}\\-\frac{i\,\lambda_2}{\rho^+}\\0\\0\\0\end{pmatrix}$$

Eigenvalue $\mu=-\frac{\lambda}{\alpha}$ **, for** $\alpha=u_1^+$

$$X_3^{(0)}=\begin{pmatrix}0\\0\\0\\-\lambda_0\\i\,u_{1^+}\end{pmatrix},\quad X_3^{(1)}=\begin{pmatrix}0\\0\\0\\-\lambda_1\\0\end{pmatrix},\quad X_3^{(2)}=\begin{pmatrix}0\\0\\0\\-\lambda_2\\0\end{pmatrix}$$

When $\lambda_0=0$ the vectors above become

$$X_3^{(0)}(\lambda_0=0)=\begin{pmatrix}0\\0\\0\\0\\i\,u_{1^+}\end{pmatrix},\quad X_3^{(1)}(\lambda_0=0)=\begin{pmatrix}0\\0\\0\\-\lambda_1\\0\end{pmatrix},\quad X_3^{(2)}(\lambda_0=0)=\begin{pmatrix}0\\0\\0\\-\lambda_2\\0\end{pmatrix}$$

When $\lambda_0=\lambda_1=0$ the vectors above become

$$X_3^{(0)}(\lambda_0=\lambda_1=0)=\begin{pmatrix}0\\0\\0\\0\\i\,u_{1^+}\end{pmatrix},\quad X_3^{(1)}(\lambda_0=\lambda_1=0)=\begin{pmatrix}0\\0\\0\\0\\0\end{pmatrix},\quad X_3^{(2)}(\lambda_0=\lambda_1=0)=\begin{pmatrix}0\\0\\0\\-\lambda_2\\0\end{pmatrix}$$

Eigenvalue $\mu=1+\epsilon^2\frac{(\lambda_0+u_1)^4\rho}{2\left((\lambda_0+u_1)^2-p_\rho\right)}$

$$X_4^{(0)} = X_4^{(0)}(\lambda_0 = 0) = X_4^{(0)}(\lambda_0 = 0, \lambda_1 = 0) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad X_4^{(1)} = \begin{pmatrix} 0 \\ 0 \\ -\frac{\lambda_0^2 u_{1^-}}{u_{1^+}} - \frac{\lambda_0 u_{1^-}^2}{u_{1^+}} - \lambda_0 u_{1^-} - u_{1^-}^2 \\ i\,\lambda_0 u_{1^-} + i\,u_{1^-}^2 \\ \frac{\lambda_0^2 u_{1^-}}{u_{1^+}} + \frac{\lambda_0 u_{1^-}^2}{u_{1^+}} \end{pmatrix},$$

$$X_4^{(2)} = \begin{pmatrix} -\frac{i\,(\lambda_0+u_{1^-})^2\left((\lambda_0+u_{1^-})u_{1^-}^2+(\lambda_0-u_{1^-})p_{\rho^-}-2\left((\lambda_0+u_{1^-})u_{1^-}-p_{\rho^-}\right)u_{1^+}\right)\rho^- u_{1^-}}{\left(u_{1^+}^2-p_{\rho^+}\right)\left((\lambda_0+u_{1^-})^2-p_{\rho^-}\right)} \\ \frac{i\,(\lambda_0+u_{1^-})^2\left((\lambda_0+u_{1^-})u_{1^-}^2+(\lambda_0-u_{1^-})p_{\rho^-}-\left((\lambda_0+u_{1^-})u_{1^-}-p_{\rho^-}\right)u_{1^+}\right)\rho^- u_{1^-}}{\left((\lambda_0+u_{1^-})^2-p_{\rho^-}\right)\rho^+ u_{1^+}} \\ -\frac{2\,\lambda_0\lambda_1 u_{1^-}}{u_{1^+}} - \frac{\lambda_1 u_{1^-}^2}{u_{1^+}} - \lambda_1 u_{1^-} \\ i\,\lambda_1 u_{1^-} \\ \frac{2\,\lambda_0\lambda_1 u_{1^-}}{u_{1^+}} + \frac{\lambda_1 u_{1^-}^2}{u_{1^+}} \end{pmatrix}$$

Fifth column

$$X_5^{(0)} = \begin{pmatrix} -\frac{2\,\lambda_0\rho^- u_{1^+}}{u_{1^+}^2-p_{\rho^+}} + \frac{2\,\lambda_0\rho^+ u_{1^+}}{u_{1^+}^2-p_{\rho^+}} \\ \frac{\lambda_0\rho^-}{\rho^+} - \lambda_0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad X_5^{(1)} = \begin{pmatrix} -\frac{2\,\lambda_1\rho^- u_{1^+}}{u_{1^+}^2-p_{\rho^+}} + \frac{2\,\lambda_1\rho^+ u_{1^+}}{u_{1^+}^2-p_{\rho^+}} \\ \frac{\lambda_1\rho^-}{\rho^+} - \lambda_1 \\ \frac{i\,p_{\rho^-}\rho^-}{\gamma} - \frac{i\,p_{\rho^+}\rho^+}{\gamma} \\ 0 \\ 0 \end{pmatrix}, \quad X_5^{(2)} = \begin{pmatrix} -\frac{2\,\lambda_2\rho^- u_{1^+}}{u_{1^+}^2-p_{\rho^+}} + \frac{2\,\lambda_2\rho^+ u_{1^+}}{u_{1^+}^2-p_{\rho^+}} \\ \frac{\lambda_2\rho^-}{\rho^+} - \lambda_2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

When $\lambda_0 = 0$ the vectors above become

$$X_5^{(0)}(\lambda_0 = 0) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad X_5^{(1)}(\lambda_0 = 0) = \begin{pmatrix} -\frac{2\,\lambda_1\rho^- u_{1^+}}{u_{1^+}^2-p_{\rho^+}} + \frac{2\,\lambda_1\rho^+ u_{1^+}}{u_{1^+}^2-p_{\rho^+}} \\ \frac{\lambda_1\rho^-}{\rho^+} - \lambda_1 \\ \frac{i\,p_{\rho^-}\rho^-}{\gamma} - \frac{i\,p_{\rho^+}\rho^+}{\gamma} \\ 0 \\ 0 \end{pmatrix}, \quad X_5^{(2)}(\lambda_0 = 0) = \begin{pmatrix} -\frac{2\,\lambda_2\rho^- u_{1^+}}{u_{1^+}^2-p_{\rho^+}} + \frac{2\,\lambda_2\rho^+ u_{1^+}}{u_{1^+}^2-p_{\rho^+}} \\ \frac{\lambda_2\rho^-}{\rho^+} - \lambda_2 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

When $\lambda_0 = \lambda_1 = 0$ the vectors above become

$$X_5^{(0)}(\lambda_0 = \lambda_1 = 0) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad X_5^{(1)}(\lambda_0 = \lambda_1 = 0) = \begin{pmatrix} 0 \\ 0 \\ \frac{i\,p_{\rho^-}\rho^-}{\gamma} - \frac{i\,p_{\rho^+}\rho^+}{\gamma} \end{pmatrix}, \quad X_5^{(2)}(\lambda_0 = \lambda_1 = 0) = \begin{pmatrix} -\frac{2\,\lambda_2\rho^- u_{1^+}}{u_{1^+}^2-p_{\rho^+}} + \frac{2\,\lambda_2\rho^+ u_{1^+}}{u_{1^+}^2-p_{\rho^+}} \\ \frac{\lambda_2\rho^-}{\rho^+} - \lambda_2 \\ 0 \end{pmatrix}$$

Lopatinski determinant of order ϵ^2.

If we take into account [table 1] we see that the only possible expression for Δ_2 is

$$\Delta_2 = det(X_1^{(1)}, X_2^{(0)}, X_3^{(0)}, X_4^{(1)}, X_5^{(0)})$$

where

$$(X_1^{(1)}, X_2^{(0)}, X_3^{(0)}, X_4^{(1)}, X_5^{(0)}) = \left(\begin{array}{c|c|c|c|c} 0 & A_1 & 0 & 0 & -\frac{2\,\lambda_0\rho^- u_{1^+}}{u_{1^+}^2-p_{\rho^+}} + \frac{2\,\lambda_0\rho^+ u_{1^+}}{u_{1^+}^2-p_{\rho^+}} \\ 0 & \frac{i\,\lambda_0^3}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)^2\rho^+} - \frac{i\,\lambda_0}{\rho^+} & 0 & 0 & \frac{\lambda_0\rho^-}{\rho^+} - \lambda_0 \\ -\lambda_0^2 + 2\,\lambda_0 u_{1^+} - u_{1^+}^2 & 0 & 0 & -\frac{\lambda_0^2 u_{1^-}}{u_{1^+}} - \frac{\lambda_0 u_{1^-}^2}{u_{1^+}} - \lambda_0 u_{1^-} - u_{1^-}^2 & 0 \\ i\,\lambda_0 u_{1^+} - i\,u_{1^+}^2 & 0 & -\lambda_0 & i\,\lambda_0 u_{1^-} + i\,u_{1^-}^2 & 0 \\ \lambda_0^2 - \lambda_0 u_{1^+} & 0 & i\,u_{1^+} & \frac{\lambda_0^2 u_{1^-}}{u_{1^+}} + \frac{\lambda_0 u_{1^-}^2}{u_{1^+}} & 0 \end{array}\right)$$

where

$$A_1 = \frac{i\,\lambda_0 u_{1^+}^2}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)p_{\rho^+}} + \frac{i\,\lambda_0^3 u_{1^+}}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)^2 p_{\rho^+}} - \frac{i\,\lambda_0^3 u_{1^+}^2}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)^3 p_{\rho^+}} - \frac{i\,\lambda_0 u_{1^+}}{p_{\rho^+}}$$

The block structure of this matrix allow us to see right away that $det(X_1^{(1)}, X_2^{(0)}, X_3^{(0)}, X_4^{(1)}, X_5^{(0)})$ is $\mathcal{O}(\lambda_0^3)$ and $o(\lambda_0^2)$. However, the $\lambda_0 \neq 0$ that provides a zero determinant implies that the eigenvalues collide. As we want to prevent glancing modes we must take $\lambda_0 = 0$. We implement this part in

In [23]:
```
#-------------------------------------------------------------------------------------------
#   Second order Lop determinant
#-------------------------------------------------------------------------------------------

Lopatinski_order2_1 = block_matrix(1,5,[X_1_one,X_2_zero,X_3_zero, X_4_one, X_5_zero]);
```

Lopatinski determinant of order ϵ^3.

From multilinearity of the determinant function and the [table 1], it is not hard to see that Δ_3 is automatically 0.

Lopatinski determinant of order ϵ^4.

Using the multilinearity of the determinant function (Remark 11) and (12), it is not hard to see that

$$\Delta_4 = det(X_1^{(1)}, X_2^{(1)}, X_3^{(0)}, X_4^{(1)}, X_5^{(1)}). \tag{13}$$

where

$$det(X_1^{(1)}, X_2^{(1)}, X_3^{(0)}, X_4^{(1)}, X_5^{(1)}) = \begin{pmatrix} 0 & \frac{i\,\lambda_1 u_{1+}^2}{\left(u_{1+}+\sqrt{p_{\rho^+}}\right)p_{\rho^+}} - \frac{i\,\lambda_1 u_{1+}}{p_{\rho^+}} & 0 & 0 & -\frac{2\,\lambda_1\rho^- u_{1+}}{u_{1+}^2-p_{\rho^+}} + \frac{2\,\lambda_1\rho^+ u_{1+}}{u_{1+}^2-p_{\rho^+}} \\ 0 & -\frac{i\,\lambda_1}{\rho^+} & 0 & 0 & \frac{\lambda_1\rho^-}{\rho^+} - \lambda_1 \\ -u_{1^+}^2 & 0 & 0 & -u_{1^-}^2 & \frac{i\,p_{\rho^-}\rho^-}{\gamma} - \frac{i\,p_{\rho^+}\rho^+}{\gamma} \\ -i\,u_{1^+}^2 & 0 & 0 & i\,u_{1^-}^2 & 0 \\ 0 & 0 & i\,u_{1^+} & 0 & 0 \end{pmatrix} \tag{2}$$

One can show that $det(X_1^{(1)}, X_2^{(1)}, X_3^{(0)}, X_4^{(1)}, X_5^{(1)})$ is$\mathcal{O}(\lambda_1^2)$ and $o(\lambda_1)$, but not $o(\lambda_1^2)$. It implies that $\lambda_1 = 0$.

In [24]:
```
#-------------------------------------------------------------------------------------------
#   Fourth order Lop determinant
#-------------------------------------------------------------------------------------------

Lopatinski_order4_1 = block_matrix(1,5,[X_1_one,X_2_one,X_3_zero, X_4_one, X_5_one]);
Lopatinski_order4_1 = Lopatinski_order4_1(lambda_0=0);
```

Lopatinski determinant of order ϵ^5.

As before [3], we will explore multilinearity to compute the fourth order term of Δ. We have only one term to check

$$\Delta_5 = det(X_1^{(1)}, X_2^{(2)}, X_3^{(0)}, X_4^{(1)}, X_5^{(1)})$$

where

$$(X_1^{(1)}, X_2^{(2)}, X_3^{(0)}, X_4^{(1)}, X_5^{(1)}) = \begin{pmatrix} 0 & \frac{i\,\lambda_2 u_{1+}^2}{\left(u_{1+}+\sqrt{p_{\rho^+}}\right)p_{\rho^+}} - \frac{i\,\lambda_2 u_{1+}}{p_{\rho^+}} & 0 & 0 & 0 \\ 0 & -\frac{i\,\lambda_2}{\rho^+} & 0 & 0 & 0 \\ -u_{1^+}^2 & 0 & 0 & -u_{1^-}^2 & \frac{i\,p_{\rho^-}\rho^-}{\gamma} - \frac{i\,p_{\rho^+}\rho^+}{\gamma} \\ -i\,u_{1^+}^2 & 0 & 0 & i\,u_{1^-}^2 & 0 \\ 0 & 0 & i\,u_{1^+} & 0 & 0 \end{pmatrix} \tag{14}$$

and, since the first two rows are linearly dependent, we can see that the determinant of the later matrix is zero.

In [25]:
```
#-------------------------------------------------------------------------------------------
#   Fifth order Lop determinant
#-------------------------------------------------------------------------------------------

Lopatinski_order5_1 = block_matrix(1,5,[X_1_one,X_2_two,X_3_zero, X_4_one, X_5_one]);
Lopatinski_order5_1 = Lopatinski_order5_1(lambda_0=0, lambda_1=0);
```

Lopatinski determinant of order ϵ^6 .

As before,Remark 11 we will explore multilinearity to compute the fourth order term of Δ. Using (13) we have

$$\Delta_6 = det(X_1^{(2)}, X_2^{(2)}, X_3^{(0)}, X_4^{(1)}, X_5^{(1)}) + det(X_1^{(1)}, X_2^{(2)}, X_3^{(0)}, X_4^{(2)}, X_5^{(1)}) + det(X_1^{(1)}, X_2^{(2)}, X_3^{(0)}, X_4^{(1)}, X_5^{(2)}) + det(X_1^{(1)}, X_2^{(3)}, X_3^{(0)}, X_4^{(1)}, X_5^{(1)})$$

It is not hard to see that the last determinant is zero (just look at the structure of columns 1, 3, 4 and 5 in (14)). For the other cases, we have:

$$(X_1^{(2)},X_2^{(2)},X_3^{(0)},X_4^{(1)},X_5^{(1)}) = \left(\begin{array}{cc|c|c|c} -\frac{i\,\rho^+ u_{1^+}^4}{u_{1^+}^2-p_{\rho^+}} & -\frac{i\,\lambda_2 u_{1^+}}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)\sqrt{p_{\rho^+}}} & 0 & 0 & 0 \\ 0 & -\frac{i\,\lambda_2}{\rho^+} & 0 & 0 & 0 \\ 0 & 0 & 0 & -u_{1^-}^2 & \frac{i\,p_{\rho^-}\rho^-}{\gamma}-\frac{i\,p_{\rho^+}\rho^+}{\gamma} \\ 0 & 0 & 0 & i\,u_{1^-}^2 & 0 \\ 0 & 0 & i\,u_{1^+} & 0 & 0 \end{array}\right) \tag{15}$$

To emphasize the block structure we will write the second matrix in a different order of the columns[4]:

$$(X_2^{(2)},X_4^{(2)},X_1^{(1)},X_3^{(0)},X_5^{(1)}) = \left(\begin{array}{cc|cc|c} -\frac{i\,\lambda_2 u_{1^+}}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)\sqrt{p_{\rho^+}}} & -\frac{i\left(u_{1^-}^3-2\left(u_{1^-}^2-p_{\rho^-}\right)u_{1^+}-p_{\rho^-}u_{1^-}\right)\rho^- u_{1^-}^3}{\left(u_{1^+}^2-p_{\rho^+}\right)\left(u_{1^-}^2-p_{\rho^-}\right)} & 0 & 0 & 0 \\ -\frac{i\,\lambda_2}{\rho^+} & \frac{i\left(u_{1^-}^3-\left(u_{1^-}^2-p_{\rho^-}\right)u_{1^+}-p_{\rho^-}u_{1^-}\right)\rho^- u_{1^-}^3}{\left(u_{1^-}^2-p_{\rho^-}\right)\rho^+ u_{1^+}} & 0 & 0 & 0 \\ 0 & 0 & -u_{1^+}^2 & 0 & \frac{i\,p_{\rho^-}\rho^-}{\gamma}-\frac{i\,p_{\rho^+}\rho^+}{\gamma} \\ 0 & 0 & -i\,u_{1^+}^2 & 0 & 0 \\ 0 & 0 & 0 & i\,u_{1^+} & 0 \end{array}\right) \tag{16}$$

As $det(X_1^{(1)},X_2^{(2)},X_3^{(0)},X_4^{(1)},X_5^{(2)}) = det(X_2^{(2)},X_5^{(2)},X_1^{(1)},X_3^{(0)},X_4^{(1)})$ we write the third matrix in {???} as

$$(X_2^{(2)},X_5^{(2)},X_1^{(1)},X_3^{(0)},X_4^{(1)}) = \left(\begin{array}{cc|c|c|c} -\frac{i\,\lambda_2 u_{1^+}}{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)\sqrt{p_{\rho^+}}} & -\frac{2\left(\rho^--\rho^+\right)\lambda_2 u_{1^+}}{u_{1^+}^2-p_{\rho^+}} & 0 & 0 & 0 \\ -\frac{i\,\lambda_2}{\rho^+} & \frac{\lambda_2\rho^-}{\rho^+}-\lambda_2 & 0 & 0 & 0 \\ 0 & 0 & -u_{1^+}^2 & 0 & -u_{1^-}^2 \\ 0 & 0 & -i\,u_{1^+}^2 & 0 & i\,u_{1^-}^2 \\ 0 & 0 & 0 & i\,u_{1^+} & 0 \end{array}\right) \tag{17}$$

Now we will use the Rankine Hugoniot conditions to write these matrices only in terms of u_1^-, ρ^- and other constants: following the notation in [FT, section 3.1],

$$R=\frac{\rho^+}{\rho^-}=\frac{u_1^-}{u_1^+}, \qquad M^2=\frac{(u_1^+)^2}{p_{\rho^+}}, \quad (M^-)^2=\frac{(u_1^-)^2}{p_{\rho}^-}=R^{\gamma+1}M^2, \qquad M^2=\frac{R^\gamma-1}{\gamma R^\gamma(R-1)}$$

so we can rewrite matrices (15), (16) and (17) as

$$(X_1^{(2)},X_2^{(2)},X_3^{(0)},X_4^{(1)},X_5^{(1)}) = \left(\begin{array}{cc|c|c|c} -\frac{i\,R\rho^- u_{1^+}^4}{u_{1^+}^2-\frac{u_{1^+}^2}{M^2}} & -\frac{i\,\lambda_2 u_{1^+}}{\left(u_{1^+}+\sqrt{\frac{u_{1^+}^2}{M^2}}\right)\sqrt{\frac{u_{1^+}^2}{M^2}}} & 0 & 0 & 0 \\ 0 & -\frac{i\,\lambda_2}{R\rho^-} & 0 & 0 & 0 \\ 0 & 0 & 0 & -R^2u_{1^+}^2 & -\frac{i\,R\rho^- u_{1^+}^2}{M^2\gamma}+\frac{i\,\rho^- u_{1^+}^2}{M^2\gamma R^{(\gamma-1)}} \\ 0 & 0 & 0 & i\,R^2u_{1^+}^2 & 0 \\ 0 & 0 & i\,u_{1^+} & 0 & 0 \end{array}\right)$$

$$(X_2^{(2)},X_4^{(2)},X_1^{(1)},X_3^{(0)},X_5^{(1)}) = \left(\begin{array}{cc|cc|c} -\frac{i\,\lambda_2 u_{1^+}}{\left(u_{1^+}+\sqrt{\frac{u_{1^+}^2}{M^2}}\right)\sqrt{\frac{u_{1^+}^2}{M^2}}} & -\frac{i\left(R^3u_{1^+}^3-2\left(R^2u_{1^+}^2-\frac{u_{1^+}^2}{M^2R^{(\gamma-1)}}\right)u_{1^+}-\frac{Ru_{1^+}^3}{M^2R^{(\gamma-1)}}\right)R^3\rho^-u_{1^+}^3}{\left(u_{1^+}^2-\frac{u_{1^+}^2}{M^2}\right)\left(R^2u_{1^+}^2-\frac{u_{1^+}^2}{M^2R^{(\gamma-1)}}\right)} & 0 & 0 & 0 \\ -\frac{i\,\lambda_2}{R\rho^-} & \frac{i\left(R^3u_{1^+}^3-\left(R^2u_{1^+}^2-\frac{u_{1^+}^2}{M^2R^{(\gamma-1)}}\right)u_{1^+}-\frac{Ru_{1^+}^3}{M^2R^{(\gamma-1)}}\right)R^2u_{1^+}^2}{R^2u_{1^+}^2-\frac{u_{1^+}^2}{M^2R^{(\gamma-1)}}} & 0 & 0 & 0 \\ 0 & 0 & -u_{1^+}^2 & 0 & -\frac{i\,R\rho^-u_{1^+}^2}{M^2\gamma}+\frac{i\,\rho^-u_{1^+}^2}{M^2\gamma R^{(\gamma-1)}} \\ 0 & 0 & -i\,u_{1^+}^2 & 0 & 0 \\ 0 & 0 & 0 & i\,u_{1^+} & 0 \end{array}\right)$$

$$(X_2^{(2)},X_5^{(2)},X_1^{(1)},X_3^{(0)},X_4^{(1)}) = \left(\begin{array}{cc|c|c|c} -\frac{i\,\lambda_2 u_{1^+}}{\left(u_{1^+}+\sqrt{\frac{u_{1^+}^2}{M^2}}\right)\sqrt{\frac{u_{1^+}^2}{M^2}}} & \frac{2\left(R\rho^--\rho^-\right)\lambda_2 u_{1^+}}{u_{1^+}^2-\frac{u_{1^+}^2}{M^2}} & 0 & 0 & 0 \\ -\frac{i\,\lambda_2}{R\rho^-} & -\lambda_2+\frac{\lambda_2}{R} & 0 & 0 & 0 \\ 0 & 0 & -u_{1^+}^2 & 0 & -R^2u_{1^+}^2 \\ 0 & 0 & -i\,u_{1^+}^2 & 0 & i\,R^2u_{1^+}^2 \\ 0 & 0 & 0 & i\,u_{1^+} & 0 \end{array}\right)$$

After simplification of some of these coefficients we will rewrite these matrices as

$$(X_1^{(2)},X_2^{(2)},X_3^{(0)},X_4^{(1)},X_5^{(1)}) = \left(\begin{array}{c|c|c|c|c} -\frac{i\,M^2R\rho^-u_{1^+}^2}{M^2-1} & -\frac{i\,\lambda_2 u_{1^+}}{\left(u_{1^+}+\sqrt{\frac{u_{1^+}^2}{M^2}}\right)\sqrt{\frac{u_{1^+}^2}{M^2}}} & 0 & 0 & 0 \\ 0 & -\frac{i\,\lambda_2}{R\rho^-} & 0 & 0 & 0 \\ 0 & 0 & 0 & -R^2u_{1^+}^2 & -\frac{i\,R\rho^-u_{1^+}^2}{M^2\gamma}+\frac{i\,\rho^-u_{1^+}^2}{M^2\gamma R^{(\gamma-1)}} \\ 0 & 0 & 0 & i\,R^2u_{1^+}^2 & 0 \\ 0 & 0 & i\,u_{1^+} & 0 & 0 \end{array}\right)$$

$$(X_2^{(2)},X_4^{(2)},X_1^{(1)},X_3^{(0)},X_5^{(1)}) = \begin{pmatrix} -\frac{i\,M^2\lambda_2}{(M+1)u_{1^+}} & -\frac{i\,(R-2)M^2R^3\rho^-u_{1^+}^2}{M^2-1} & 0 & 0 & 0 \\ -\frac{i\,\lambda_2}{R\rho^-} & i\,(R-1)R^2u_{1^+}^3 & 0 & 0 & 0 \\ 0 & 0 & -u_{1^+}^2 & 0 & -\frac{i\,R\rho^-u_{1^+}^2}{M^2\gamma}+\frac{i\,\rho^-u_{1^+}^2}{M^2\gamma R^{(\gamma-1)}} \\ 0 & 0 & -i\,u_{1^+}^2 & 0 & 0 \\ 0 & 0 & 0 & i\,u_{1^+} & 0 \end{pmatrix}$$

$$(X_2^{(2)},X_5^{(2)},X_1^{(1)},X_3^{(0)},X_4^{(1)}) = \begin{pmatrix} -\frac{i\,M^2\lambda_2}{(M+1)u_{1^+}} & \frac{2\,(R-1)M^2\lambda_2\rho^-}{\left(M^2-1\right)u_{1^+}} & 0 & 0 & 0 \\ -\frac{i\,\lambda_2}{R\rho^-} & -\lambda_2+\frac{\lambda_2}{R} & 0 & 0 & 0 \\ 0 & 0 & -u_{1^+}^2 & 0 & -R^2u_{1^+}^2 \\ 0 & 0 & -i\,u_{1^+}^2 & 0 & i\,R^2u_{1^+}^2 \\ 0 & 0 & 0 & i\,u_{1^+} & 0 \end{pmatrix}$$

It is not hard to see that the above determinants can be easily computed using the block structure of these matrices. We will denote the upper (resp. lower) $2X2$ minor (resp. $3X3$ minor) by an subindex upper (resp. minor)

Lower $3X3$ minors computation

Using the last Rankine-Hugoniot condition in the next two cases one obtain

$$\det(X_1^{(2)},X_2^{(2)},X_3^{(0)},X_4^{(1)},X_5^{(1)})_{\text{lower}} = \left(-\frac{i\,R\rho^-u_{1^+}^2}{M^2\gamma}+\frac{i\,\rho^-u_{1^+}^2}{M^2\gamma R^{(\gamma-1)}}\right)R^2u_{1^+}^3 = -i\,(R-1)R^3\rho^-u_{1^+}^5$$

$$\det(X_2^{(2)},X_4^{(2)},X_1^{(1)},X_3^{(0)},X_5^{(1)})_{\text{lower}} = \left(-\frac{i\,R\rho^-u_{1^+}^2}{M^2\gamma}+\frac{i\,\rho^-u_{1^+}^2}{M^2\gamma R^{(\gamma-1)}}\right)u_{1^+}^3 = -i\,(R-1)R\rho^-u_{1^+}^5$$

and one can easily show that

$$\det(X_2^{(2)},X_5^{(2)},X_1^{(1)},X_3^{(0)},X_4^{(1)})_{\text{lower}} = -2\,R^2u_{1^+}^5$$

Upper $2x2$ minors computation

$$\det(X_1^{(2)},X_2^{(2)},X_3^{(0)},X_4^{(1)},X_5^{(1)})_{\text{upper}} = -\frac{M^2\lambda_2u_{1^+}^2}{M^2-1}$$

$$\det(X_2^{(2)},X_4^{(2)},X_1^{(1)},X_3^{(0)},X_5^{(1)})_{\text{upper}} = \frac{(R-2)M^2R^2\lambda_2u_{1^+}^2}{M^2-1}+\frac{(R-1)M^2R^2\lambda_2u_{1^+}^2}{M+1} = \frac{((R-1)M-1)M^2R^2\lambda_2u_{1^+}^2}{M^2-1}$$

$$\det(X_2^{(2)},X_5^{(2)},X_1^{(1)},X_3^{(0)},X_4^{(1)})_{\text{upper}} = \frac{2i\,(R-1)M^2\lambda_2^2}{\left(M^2-1\right)Ru_{1^+}}+\frac{i\left(\lambda_2-\frac{\lambda_2}{R}\right)M^2\lambda_2}{(M+1)u_{1^+}} = \frac{(i\,R-i)M^2\lambda_2^2}{(M-1)Ru_{1^+}}$$

Now we put them all together:

$$\det(X_1^{(2)},X_2^{(2)},X_3^{(0)},X_4^{(1)},X_5^{(1)}) = \frac{i\,(R-1)M^2R^3\lambda_2\rho^-u_{1^+}^7}{M^2-1}$$

$$\det(X_2^{(2)},X_4^{(2)},X_1^{(1)},X_3^{(0)},X_5^{(1)}) = -\frac{i\,(R-1)((R-1)M-1)M^2R^3\lambda_2\rho^-u_{1^+}^7}{M^2-1}$$

$$\det(X_2^{(2)},X_5^{(2)},X_1^{(1)},X_3^{(0)},X_4^{(1)}) = -\frac{2\,(i\,R-i)M^2R\lambda_2^2u_{1^+}^4}{M-1}$$

Recall that

$$\Delta_6 = det(X_1^{(2)},X_2^{(2)},X_3^{(0)},X_4^{(1)},X_5^{(1)}) - det(X_2^{(2)},X_4^{(2)},X_1^{(1)},X_3^{(0)},X_5^{(1)}) + det(X_2^{(2)},X_5^{(2)},X_1^{(1)},X_3^{(0)},X_4^{(1)})$$

we can simplify the first two terms on the right hand side to

$$det(X_1^{(2)},X_2^{(2)},X_3^{(0)},X_4^{(1)},X_5^{(1)}) - det(X_2^{(2)},X_4^{(2)},X_1^{(1)},X_3^{(0)},X_5^{(1)}) = \frac{i\,(R-1)^2M^3R^3\lambda_2\rho^-u_{1^+}^7}{M^2-1}$$

and we end up with

$$\Delta_6 = \frac{i\,(R-1)^2M^3R^3\lambda_2\rho^-u_{1^+}^7}{M^2-1} - \frac{2\,(i\,R-i)M^2R\lambda_2^2u_{1^+}^4}{M-1}$$

and then

$$\lambda_2 = \frac{MR^3\rho^- u_{1^+}^3 - MR^2\rho^- u_{1^+}^3}{2\,(M+1)} = \frac{(R-1)MR^2\rho^- u_{1^+}^3}{2\,(M+1)} \tag{18}$$

This function is clearly positive for all $R > 1$. This is much stronger than the statement in [FT, page3036] since it gives the result explicitly for all values of $\gamma \geq 1$ (!).

In the next section we compare the values for the Lopatinski root with those found numerically.

In [26]:
```
## WE begin by simplifying two columns a little bit
Y_2_two= X_2_two(lambda_0=0,lambda_1=0);
Y_2_two = Y_2_two.substitute(I*lambda_2*u_1_plus^2/((u_1_plus + sqrt(p_rho_plus))*p_rho_plus) - I*lambda_2*u_1_plus/p_rho
_plus ==-i*lambda_2*u_1_plus/((u_1_plus + sqrt(p_rho_plus))*sqrt(p_rho_plus))); ##OK

Y_1_two = X_1_two(lambda_0=0,lambda_1 =0);
Y_1_two = Y_1_two.substitute(-I*rho_plus*u_1_plus^6/((u_1_plus^2 - p_rho_plus)*p_rho_plus) + I*rho_plus*u_1_plus^4/p_rho_
plus ==  -I*rho_plus*u_1_plus^4/(u_1_plus^2 - p_rho_plus) ); ##OK

Y_5_two = X_5_two(lambda_0=0,lambda_1 =0);
Y_5_two = Y_5_two.substitute(-2*lambda_2*rho_minus*u_1_plus/(u_1_plus^2 - p_rho_plus) + 2*lambda_2*rho_plus*u_1_plus/(u_1
_plus^2 - p_rho_plus) ==
                                2*lambda_2*u_1_plus*(rho_plus - rho_minus)/(u_1_plus^2 - p_rho_plus)); ##OK

### And then we compute the lopatinski Determinant

Lopatinski_order6_1= block_matrix(1,5,[Y_1_two,Y_2_two,X_3_zero, X_4_one, X_5_one]);
Lopatinski_order6_1 = Lopatinski_order6_1(lambda_0=0, lambda_1=0);

Lopatinski_order6_2= block_matrix(1,5,[Y_2_two, X_4_two,X_1_one,X_3_zero, X_5_one]); ### It HAS a minus sign in front!!!
Lopatinski_order6_2 = Lopatinski_order6_2(lambda_0=0, lambda_1=0);

Lopatinski_order6_3= block_matrix(1,5,[Y_2_two,Y_5_two, X_1_one,  X_3_zero, X_4_one]);   ### It DOESN'T HAVE a minus sign
 in front!!!
Lopatinski_order6_3 = Lopatinski_order6_3(lambda_0=0, lambda_1=0);

#### SUbstitutions

var('R M');

#rho_plus  \to  R*rho_minus;
#u_1_minus \to R*u_1_plus;
#p_rho_plus  \to (u_1_plus/M)^2


Lopatinski_order6_1 = Lopatinski_order6_1.substitute(rho_plus ==R*rho_minus);
Lopatinski_order6_1 = Lopatinski_order6_1.substitute(u_1_minus == R*u_1_plus);
Lopatinski_order6_1 = Lopatinski_order6_1.substitute(p_rho_plus == (u_1_plus/M)^2);
Lopatinski_order6_1 = Lopatinski_order6_1.substitute(p_rho_minus == u_1_plus^2/(M^2*R^(gamm-1)) );

Lopatinski_order6_2 = Lopatinski_order6_2.substitute(rho_plus ==R*rho_minus);
Lopatinski_order6_2 = Lopatinski_order6_2.substitute(u_1_minus == R*u_1_plus);
Lopatinski_order6_2 = Lopatinski_order6_2.substitute(p_rho_plus == (u_1_plus/M)^2);
Lopatinski_order6_2 = Lopatinski_order6_2.substitute(p_rho_minus == u_1_plus^2/(M^2*R^(gamm-1)) );

Lopatinski_order6_3 = Lopatinski_order6_3.substitute(rho_plus ==R*rho_minus);
Lopatinski_order6_3 = Lopatinski_order6_3.substitute(u_1_minus == R*u_1_plus);
Lopatinski_order6_3 = Lopatinski_order6_3.substitute(p_rho_plus == (u_1_plus/M)^2);
Lopatinski_order6_3 = Lopatinski_order6_3.substitute(p_rho_minus == u_1_plus^2/(M^2*R^(gamm-1)) );


#Short break for a test :)

#R = u_1_minus/u_1_plus;
#R = R.substitute(u_1_minus==1);
#R = R.substitute( u_1_plus==.4);
#M = sqrt((R^gamm -1)/(gamm*R^(gamm)*(R-1)))
##M = M.substitute(R ==5);
#M = M.substitute( gamm == 5/3);


#Lop1 = Lopatinski_order6_1.substitute(epsilon = 1/2, gamm = 5/3, R=2.5, M = 0.559587916478692, u_1_plus=.4,rho_minus=1)
#Lop2 = Lopatinski_order6_2.substitute(epsilon = 1/2, gamm = 5/3, R=2.5, M = 0.559587916478692, u_1_plus=.4,rho_minus=1)
#Lop3 = Lopatinski_order6_3.substitute(epsilon = 1/2, gamm = 5/3, R=2.5, M = 0.559587916478692, u_1_plus=.4,rho_minus=1)
#lop1 = Lop1.determinant();
#lop2 = Lop2.determinant();
#lop3 = Lop3.determinant();
```

```
#pol = lop1 - lop2 + lop3;


# TIME TO SIMPLIFY EXPRESSIONS!!

# We will simplify  Lopatinski_order6_2[0,1] , Lopatinski_order6_2[1,1] ,  Lopatinski_order6_3[0,0] , Lopatinski_order6_3
[0,1];
# This is enough for simplifying our computations considerably


###HUMAN INTERVENTION
Lopatinski_order6_1[0,0] = -I*R*rho_minus*u_1_plus^2*M^2/(M^2 -1);   # OK
#w = -I*R*rho_minus*u_1_plus^4/(u_1_plus^2 - u_1_plus^2/M^2);
#w = -w -I*R*rho_minus*u_1_plus^2*M^2/(M^2 -1);
#w.substitute(R = 2, rho_minus=.3, M=5, gamm = 1.4, u_1_plus=.32)

Lopatinski_order6_2[0,0] = -I*lambda_2*M^2/(u_1_plus*(M+1)); #OK
#w = -I*lambda_2*u_1_plus/((u_1_plus + sqrt(u_1_plus^2/M^2))*sqrt(u_1_plus^2/M^2));
#w = -w -I*lambda_2*M^2/(u_1_plus*(M+1));

Lopatinski_order6_2[0,1] = -i*u_1_plus^2*(R-2)*R^3*rho_minus*M^2/(M^2 -1);   #OK
#w= -I*(R^3*u_1_plus^3 - 2*(R^2*u_1_plus^2 - u_1_plus^2/(M^2*R^(gamm - 1)))*u_1_plus - R*u_1_plus^3/(M^2*R^(gamm - 1)))*R
^3*rho_minus*u_1_plus^3/((u_1_plus^2 - u_1_plus^2/M^2)*(R^2*u_1_plus^2 - u_1_plus^2/(M^2*R^(gamm - 1))));
#w = -w -i*u_1_plus^2*(R-2)*R^3*rho_minus*M^2/(M^2 -1);


Lopatinski_order6_2[1,1] = i*R^2*u_1_plus^3*(R-1);    #OK
#w = I*(R^3*u_1_plus^3 - (R^2*u_1_plus^2 - u_1_plus^2/(M^2*R^(gamm - 1)))*u_1_plus - R*u_1_plus^3/(M^2*R^(gamm - 1)))*R^2
*u_1_plus^2/(R^2*u_1_plus^2 - u_1_plus^2/(M^2*R^(gamm - 1)))

#w = -w +i*R^2*u_1_plus^3*(R-1);


Lopatinski_order6_3[0,0] = -I*lambda_2*M^2/(u_1_plus*(M+1));   #OK
#w = -I*lambda_2*u_1_plus/((u_1_plus + sqrt(u_1_plus^2/M^2))*sqrt(u_1_plus^2/M^2))
#w = -w -I*lambda_2*M^2/(u_1_plus*(M+1));

Lopatinski_order6_3[0,1] = 2*rho_minus*lambda_2*M^2*(R - 1)/(u_1_plus*(M^2 - 1));     #OK
#w = 2*(R*rho_minus - rho_minus)*lambda_2*u_1_plus/(u_1_plus^2 - u_1_plus^2/M^2);
#w = -w +2*rho_minus*lambda_2*M^2*(R - 1)/(u_1_plus*(M^2 - 1));


#w = -I*lambda_2*u_1_plus/((u_1_plus + sqrt(u_1_plus^2/M^2))*sqrt(u_1_plus^2/M^2));
#w = -w  -I*lambda_2*M^2/(u_1_plus*(M+1));
#w.substitute(R = 2, rho_minus=.3, M=5, gamm = 1.4, u_1_plus=.32)


#### Lower and Upper minors   ### The main concern here is that I used a RH formula as provided by FT... so far it is not
 veryfing numerically. Let's try again :|
### IT WORKED!!!! THERE WAS A MISTAKE ON MY RANKINE HUGONIOT CONDITIONS!!!

##Lower 3x3 blocks
Lopatinski_order6_1_lower = Lopatinski_order6_1[(2,3,4),(2,3,4)];
detLopatinski_order6_1_lower = Lopatinski_order6_1_lower.determinant();
detLopatinski_order6_1_lower = detLopatinski_order6_1_lower.substitute((-I*R*rho_minus*u_1_plus^2/(M^2*gamm) + I*rho_minu
s*u_1_plus^2/(M^2*gamm*R^(gamm - 1)))== -I*R*(R-1)*rho_minus*u_1_plus^2);

Lopatinski_order6_2_lower = Lopatinski_order6_2[(2,3,4),(2,3,4)];
detLopatinski_order6_2_lower = Lopatinski_order6_2_lower.determinant();
detLopatinski_order6_2_lower = detLopatinski_order6_2_lower.substitute((-I*R*rho_minus*u_1_plus^2/(M^2*gamm) + I*rho_minu
s*u_1_plus^2/(M^2*gamm*R^(gamm - 1)))== -I*R*(R-1)*rho_minus*u_1_plus^2);

#w =(-I*R*rho_minus*u_1_plus^2/(M^2*gamm) + I*rho_minus*u_1_plus^2/(M^2*gamm*R^(gamm - 1)));   ### VERIFY!!!
#w = -w -I*R*(R-1)*rho_minus*u_1_plus^2;
#w.substitute( gamm = 5/3 ,  R=2.5, M = 0.559587916478692, u_1_plus=.4,rho_minus=1)
#w.substitute(gamm = 5/3, R=5, M = 0.37382, u_1_plus=.2,rho_minus=1) #0.037223

Lopatinski_order6_3_lower = Lopatinski_order6_3[(2,3,4),(2,3,4)];
detLopatinski_order6_3_lower = Lopatinski_order6_3_lower.determinant();

##Upper 2X2 blocks
Lopatinski_order6_1_upper = Lopatinski_order6_1[(0,1),(0,1)];
detLopatinski_order6_1_upper = Lopatinski_order6_1_upper.determinant();

Lopatinski_order6_2_upper = Lopatinski_order6_2[(0,1),(0,1)];
detLopatinski_order6_2_upper = Lopatinski_order6_2_upper.determinant();
detLopatinski_order6_2_upper = detLopatinski_order6_2_upper.substitute((R - 2)*M^2*R^2*lambda_2*u_1_plus^2/(M^2 - 1) + (R
 - 1)*M^2*R^2*lambda_2*u_1_plus^2/(M + 1) == M^2*R^2*lambda_2*u_1_plus^2*(M*(R-1) -1)/(M^2 - 1) )     #OK

#w = (R - 2)*M^2*R^2*lambda_2*u_1_plus^2/(M^2 - 1) + (R - 1)*M^2*R^2*lambda_2*u_1_plus^2/(M + 1);
#w = -w + M^2*R^2*lambda_2*u_1_plus^2*(M*(R-1) -1)/(M^2 - 1);
#w.substitute( gamm = 5/3 ,  R=2.5, M = 0.559587916478692, u_1_plus=.4,rho_minus=1)

Lopatinski_order6_3_upper = Lopatinski_order6_3[(0,1),(0,1)];
detLopatinski_order6_3_upper = Lopatinski_order6_3_upper.determinant();
#detLopatinski_order6_3_upper = detLopatinski_order6_3_upper.substitute(2*I*(R - 1)*M^2*lambda_2^2/((M^2 - 1)*R*u_1_plus)
```

```
 + I*(lambda_2 - lambda_2/R)*M^2*lambda_2/((M + 1)*u_1_plus)  == I*(R - 1)*M^2*lambda_2^2/(u_1_plus*R*(M-1)));
detLopatinski_order6_3_upper =  I*(R - 1)*M^2*lambda_2^2/(u_1_plus*R*(M-1));   #OK

#w = 2*I*(R - 1)*M^2*lambda_2^2/((M^2 - 1)*R*u_1_plus) + I*(lambda_2 - lambda_2/R)*M^2*lambda_2/((M + 1)*u_1_plus);
#w = -w  +I*(R - 1)*M^2*lambda_2^2/(u_1_plus*R*(M-1));
#w.substitute( gamm = 5/3 ,  R=2.5, M = 0.559587916478692, u_1_plus=.4,rho_minus=1)
# Now divide every lower minor by R^2*u_1_plus^5

#detLopatinski_order6_1_lower =              detLopatinski_order6_1_lower/(R^2*u_1_plus^5);
#detLopatinski_order6_2_lower =              detLopatinski_order6_2_lower/(R^2*u_1_plus^5);
#detLopatinski_order6_3_lower =              detLopatinski_order6_3_lower/(R^2*u_1_plus^5);
```

Comparison: analytical vs numerical results

As shown before, the root λ of the Lopatinski determinant is given by

$$\lambda(\epsilon) = \lambda_2\epsilon^2 + \mathcal{O}(\epsilon^4)$$

where λ_2 is given by (18). The collumn "expansion zero" denotes the roots found by pluging values on the lopatinki matrix and finding the roots (this is what we had done before, a few months ago, in order to compare the numerics and the analysis).

In [27]:
```
a_1 = detLopatinski_order6_1_lower*detLopatinski_order6_1_upper;
a_2 = detLopatinski_order6_2_lower*detLopatinski_order6_2_upper;
a_3 = detLopatinski_order6_3_lower*detLopatinski_order6_3_upper;
aux = a_1 - a_2 + a_3;

aux = I*(R - 1)^2*M^3*R^3*lambda_2*rho_minus*u_1_plus^7/(M^2 - 1) + a_3;

aux = aux.roots(lambda_2)[0][0];
aux = 1/2*(M*R^3*rho_minus*u_1_plus^3 - M*R^2*rho_minus*u_1_plus^3)/(M + 1);
aux = 1/2*(M*R^2*rho_minus*u_1_plus^3*(R-1))/(M + 1);
```

For instance, if we set $\epsilon = 1/2, \gamma = 5/3, u_1^+ = .4$ and $u_1^- = 1$ and $\rho^- = 1$ we have

$R = 2.5$ and $M = 0.559587916478692$, which gives

In [28]:
```
aux.substitute(epsilon = 1/2, gamm = 5/3, R=2.5, M = 0.559587916478692, u_1_plus=.4,rho_minus=1) #  0.107641495019176  WO
RKS!!!!
```

Out[28]:
```
0.107641495019176
```

SImilarly, if we set $\epsilon = 1/2, \gamma = 5/3, u_1^+ = .2$ and $u_1^- = 1$ and $\rho^- = 1$ we have

$R = 5$ and $M = 0.37382$, which gives

In [29]:
```
aux.substitute(epsilon = 1/2, gamm = 5/3, R=5, M = 0.37382, u_1_plus=.2,rho_minus=1) # 0.108841041766753
```

Out[29]:
```
0.108841041766753
```

which we will use to compare with the numerical results.

Case1 ($u_1^+ = 0.2, \rho_+ = 1/0.2, u_1^- = 1, \rho_- = 1$ and $\gamma = 5/3$)

In this case, we have

$$R = 5 \quad M = 0.37382$$

Hence $\lambda_2 = 0.108841041766753$.

ϵ	Numerical zero	Expansion zero	Analytical result	relative error (Numerical vs expansion zero)
1/2	0.0274	0.027210	0.027210	0.0087
1/4	0.006826	0.0068025	0.0068026	0.0034
1/8	0.001702	0.0017006	0.0017006	9.46e-4
1/16	4.252	4.2516e-04	4.2516e-04	1.47e-4
1/32	1.0627e-4	1.0629e-04	1.0629e-04	1.1e-4

We compare the zeros of the Lopatinski determinant as determined numerically and from the expansion. Here $\epsilon := 1/h_1$ and $\gamma = 5/3$ and $u_1^+ = 0.2, \rho_+ = 1/0.2$. The zeros were found numerically by using the method of moments on a circle centered at approximately the root using 10,000 contour points. By the time $\epsilon = 1/32$, I think we are getting some numerical error that is making it difficult to solve for the roots numerically.

Case 2 (For $u_1^+ = 0.4, \rho_+ = 1/0.4, u_1^- = 1, \rho_- = 1$)

In this case we have

$$R = 2.5 \quad M = 0.559587916478691$$

Hence $\lambda_2 = 0.107641495019176$.

ϵ	Numerical zero	Expansion zero	Analytical result	relative error (Numerical vs expansion zero)
1/2	0.02696	0.026910	0.026910	0.002
1/4	0.00674	0.0067276	0.0067276	0.0015
1/8	0.001683	0.0016819	0.0016819	3.97e-4
1/16	4.2048	4.2047e-04	4.2047e-04	2.64e-5
1/32	1.05104	1.0512e-04	1.0512e-04	1.50e-4

We compare the zeros of the Lopatinski determinant as determined numerically and from the expansion. Here $\epsilon := 1/h_1$ and $\gamma = 5/3$ and $u_1^+ = 0.4, \rho_+ = 1/0.4$. The zeros were found numerically by using the method of moments on a circle centered at approximately the root using 10,000 contour points. By the time $\epsilon = 1/32$, I think we are getting some numerical error that is making it difficult to solve for the roots numerically.

Case 3 (For $u_1^+ = 0.6$ and $\gamma = 5/3$.)

$$R = 10/6 \quad M = 0.71823$$

% Hence $\lambda_2 = 0.0836011476926837$.

ϵ	Numerical zero	Expansion zero	Analytical result	relative error (Numerical vs expansion zero)
1/2	0.02047	0.02090	0.020900	2×10^{-2}
1/4	0.005215	0.00523	0.0052251	3×10^{-3}
1/8	0.0013059	0.0013063	0.0013063	3×10^{-4}
1/16	0.00032652	0.00032657	3.2657e-04	2×10^{-5}
1/32	8.164e-05	8.1642e-05	8.1642e-05	2×10^{-5}
1/64	$\approx 2e-05$	2.04e-05		2×10^{-2}

We compare the zeros of the Lopatinski determinant as determined numerically and from the expansion. Here $\epsilon := 1/h_1$ and $\gamma = 5/3$ and $u_1^+ = 0.6$. The zeros were found numerically by using the method of moments on a circle centered at approximately the root using 10,000 contour points. By the time $\epsilon = 1/32$, I think we are getting some numerical error that is making it difficult to solve for the roots numerically.

Case 4(For $u_1^+ = 0.8, \rho_+ = 1/0.8$ and $\gamma = 5/3$)

$$R = 10/8 \quad M = 0.86336$$

% Hence $\lambda_2 = 0.0463335050661171$.

ϵ	Numerical zero	Expansion zero	Analytical result	relative error (Numerical vs expansion zero)
1/2	0.0091	0.011583	0.011583	0.267
1/4	0.00285	0.0028958	0.0028958	0.015
1/8	7.228e-4	7.2396e-04	7.2396e-04	0.0015
1/16	1.8093e-4	1.8099e-04	1.8099e-04	3.44e-4
1/32	4.52399	4.5248e-05	4.5248e-05	1.81e-4

We compare the zeros of the Lopatinski determinant as determined numerically and from the expansion. Here $\epsilon := 1/h_1$ and $\gamma = 5/3$ and $u_1^+ = 0.8, \rho_+ = 1/0.8$. The zeros were found numerically by using the method of moments on a circle centered at approximately the root using 10,000 contour points.

The case study of glancing modes

It is asserted in [FT, page 3035], that there exists a root to the Lopatinski determinant with order $\mathcal{O}(1)$, but that it consists of a glancing mode, i.e., there is a degeneracy of the eigenvectors at this point. In order to show that the Lopatinski determinant is not zero we need to use generalized eigenvectors and with them we compute the Lopatinski determinant. IT turns out then that we can begin solving this problem by finding a numerical/simbolic method to construct generalized eigenvector, which consists in solving the following problem:

> Given a pair eigenvalue/associated eigenvector $(\mu, x(\mu))$ of the matrix S find a vector b such that
> $$(S - \mu * I) * b = x(mu), \tag{19}$$
> where A and $x_m u$ are outputs of the following program:

In [30]:
```
var('alpha rho u_1 p_rho h_1 xi lambd lambd_0 lambd_1 lambd_2 mu_0 mu_1 mu_2')

A_0 = matrix([[1,0,0, 0,0],[u_1,rho, 0,0,0], [ 0,0,rho, 0,0], [0,0,0,1,0], [0,0,0,0,1]]);
A_1 = matrix([[u_1,rho,0, 0,0],[u_1*u_1 + p_rho,2*rho*u_1,0,0,0], [ 0,0,rho*u_1, 0,-h_1], [0,0,0,alpha,0], [0,0,-h_1,0,u_
1]]);
A_2 = matrix([[0,0,rho, 0,0],[0,0,rho*u_1,0,0], [ p_rho,0,0, h_1,0], [0,0,h_1,0,alpha - u_1], [0,0,0,0,0]]);
```

```
### Inverse of A_1
A_1_inv_aux = matrix([[2*rho*u_1,-rho,0, 0,0],[-(u_1*u_1 + p_rho), u_1,0,0,0], [ 0,0,u_1, 0,h_1], [0,0,0,1/alpha,0], [0,0
,h_1,0,rho*u_1]]);
a_1= 1/(rho*(u_1^2 - p_rho));
a_2 =1/(rho*u_1^2 - h_1^2);

D = diagonal_matrix([a_1,a_1, a_2,1,a_2]);
A_1_inv = D*A_1_inv_aux;

##############################################

S = lambd*A_1_inv*A_0 + i*xi*A_1_inv*A_2;
P_1 = matrix([[1,0,0,0,0],[0,1,0,0,0],[0,0,1,0,0],[0,0,0,1,0],[0,0,-h_1/u_1,0,1]]);

####Now we do P_1*S*P_1.inverse() , at equation (7)
S = P_1*S*P_1.inverse();

### Simplifying the second row
P_2 = matrix([[1,0,0,0,0],[p_rho/(rho*u_1),1,0,0,0],[0,0,1,0,0],[0,0,0,1,0],[0,0,0,0,1]]);

### ....we obtain equation (9)
S = P_2*S*P_2.inverse();

### Then we do S = P_2*S*S_2.inverse()
D = diagonal_matrix([1, 1, h_1/u_1, 1, 1]);

### Equation (10)
S = D*S*D.inverse();
S = -S;
S = S.expand();

var('x_1 x_2 x_3 x_4 x_5 num ')
var('alpha rho u_1 p_rho h_1 xi lambd lambd_0 lambd_1 lambd_2 mu_0 mu_1 mu_2 mu epsilon ')

S= S.substitute(xi==1);
S= S.substitute(-lambd*u_1/(u_1^2 - p_rho) - lambd*p_rho/((u_1^2 - p_rho)*u_1) ==-lambd*(u_1^2 +p_rho)/((u_1^2 - p_rho)*u
_1));
S= S.substitute( lambd*p_rho/((u_1^2 - p_rho)*rho) - lambd*p_rho^2/((u_1^2 - p_rho)*rho*u_1^2) == lambd*p_rho*(u_1^2 -p_r
ho)/((u_1^2 - p_rho)*rho*u_1^2) );
S= S.substitute(-lambd*rho*u_1/(rho*u_1^2 - h_1^2) - h_1^2*lambd/((rho*u_1^2 - h_1^2)*u_1) ==-lambd*(rho*u_1^2 + h_1^2)/(
(rho*u_1^2 - h_1^2)*u_1));
S= S.substitute(-lambd*rho*u_1/(rho*u_1^2 - h_1^2) + h_1^2*lambd/((rho*u_1^2 - h_1^2)*u_1)  == -lambd/u_1)
S = S.substitute(alpha == u_1);
S = S.substitute( -lambd*u_1/(u_1^2 - p_rho) + lambd*p_rho/((u_1^2 - p_rho)*u_1) == -lambd/u_1);



A = S - mu*identity_matrix(5);

A = A.substitute(alpha = u_1);
x = column_matrix([x_1,x_2, x_3, x_4,lambd/h_1]);
f(x_1,x_2, x_3, x_4,x_5)= (A[(4),(0,1,2,3,4,)]*x())[0,0];
sols = solve( [f()== 0], x_3);
a= x;
a = a.subs(sols[0]);

### x_4 must be....
f(x_1,x_2, x_3, x_4)= (A[(3),(0,1,2,3,4,)]*x())[0,0];
sols= solve( [f(x_3= a[2][0])== 0 ], x_4);
a = a.subs(sols[0]);

### x_1 must be....
f(x_1,x_2, x_3, x_4)= (A[(2),(0,1,2,3,4,)]*x())[0,0];
sols= solve( [f(x_3 = a[2][0],x_4= a[3][0])== 0], x_1);
a = a.subs(sols[0]);
a = a.substitute((-I*mu^2*rho*u_1^3 - 2*I*lambd*mu*rho*u_1^2 - (-I*h_1^2*mu^2 + I*lambd^2*rho + I*h_1^2)*u_1)/(h_1^2*p_rh
o)== ( -i*rho*u_1*(mu*u_1 + lambd)^2/(h_1^2*p_rho) + i*u_1*(mu^2 - 1)/(p_rho)));   ##OK

#### and finally, x_2 must be....
f(x_1,x_2, x_3, x_4)= (A[(1),(0,1,2,3,4,)]*x())[0,0];
sols= solve( [f(x_3 = a[2][0],x_5= a[4][0],x_1= a[0][0],x_4= a[3][0] )== 0], x_2);

x_mu= a.subs(sols[0]);
#x_mu[1] = x_mu[1].substitute((h_1*mu*rho*u_1 + h_1*lambd*rho) ==h_1*rho*(mu*u_1 +lambd) )
x_mu[1] = x_mu[1].substitute(((-I*mu^2*rho*u_1^2 + I*h_1^2*mu^2 - I*h_1^2 - 2*I*lambd*mu*rho*u_1 - I*lambd^2*rho)*lambd/(
(h_1*mu*rho*u_1 + h_1*lambd*rho))) == lambd*(-i*rho*(mu*u_1 +lambd)^2 + i*h_1^2*(mu^2 -1))/(h_1*rho*(mu*u_1 + lambd))); #
#OK
x_mu = x_mu.substitute(((-I*lambd*mu^2*rho*u_1^2 + I*h_1^2*lambd*mu^2 - I*h_1^2*lambd - 2*I*lambd^2*mu*rho*u_1 - I*lambd^
3*rho)/((h_1^2*mu*rho*u_1 + h_1^2*lambd*rho))) == (-I*lambd*rho*(mu*u_1+ lambd)/(h_1^2*rho)  + I*lambd*(mu^2 -1)/(rho*(mu
*u_1+ lambd)) )) ; ##OK

x_mu = (mu*u_1+lambd)*x_mu;

x_mu[1] = (I*(mu^2 - 1)*lambd/rho - I*(mu*u_1 + lambd)^2*lambd/h_1^2) ;
```

$$A = S - \mu I = \begin{pmatrix} -\mu - \frac{(u_1^2+p_\rho)\lambda}{(u_1^2-p_\rho)u_1} & \frac{\lambda\rho}{u_1^2-p_\rho} & -\frac{i\,\rho u_1^2}{(u_1^2-p_\rho)h_1} & 0 & 0 \\ \frac{\lambda p_\rho}{\rho u_1^2} & -\mu - \frac{\lambda}{u_1} & 0 & 0 & 0 \\ -\frac{i\,h_1 p_\rho}{\rho u_1^2 - h_1^2} & 0 & -\mu - \frac{(\rho u_1^2+h_1^2)\lambda}{(\rho u_1^2-h_1^2)u_1} & -\frac{i\,h_1^2}{\rho u_1^2-h_1^2} & -\frac{h_1^2\lambda}{(\rho u_1^2-h_1^2)u_1} \\ 0 & 0 & -i & -\mu - \frac{\lambda}{u_1} & 0 \\ 0 & 0 & -\frac{\lambda}{u_1} & 0 & -\mu - \frac{\lambda}{u_1} \end{pmatrix}, \quad \text{and} \quad x\_mu = \begin{pmatrix} \left(\frac{i\,(\mu^2-1)u_1}{p_\rho} - \frac{i\,(\mu u_1+\lambda)^2\rho u_1}{h_1^2 p_\rho}\right)(\mu u_1+\lambda) \\ \frac{(i\,\mu^2-i)\lambda}{\rho} - \frac{i\,(\mu u_1+\lambda)^2\lambda}{h_1^2} \\ -\frac{(\mu u_1+\lambda)^2}{h_1} \\ \frac{i\,(\mu u_1+\lambda)u_1}{h_1} \\ \frac{(\mu u_1+\lambda)\lambda}{h_1} \end{pmatrix}$$

Let's take a vector b of the form

$$b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix}$$

Let's try to solve the system (19) as a function of b_5 :

In [31]:
```
var('b_1 b_2 b_3 b_4 b_5');

### We will consider b_5 as a dummy variable
### In this case, b_3 must be....

b = column_matrix([b_1,b_2, b_3, b_4,b_5]);
f(b_1,b_2, b_3, b_4,b_5)= (A[(4),(0,1,2,3,4,)]*b())[0,0];
sols = solve( [f()== x_mu[4,0]], b_3);
b = b.subs(sols[0]);

### ... b_4 must be....
f(b_1,b_2, b_3, b_4,b_5)= (A[(3),(0,1,2,3,4,)]*b())[0,0];
sols= solve( [f()== x_mu[3,0] ], b_4);
b = b.subs(sols[0]);

### b_1 must be....
f(b_1,b_2, b_3, b_4,b_5)= (A[(2),(0,1,2,3,4,)]*b())[0,0];
sols= solve( [f()== x_mu[2,0]], b_1);
b = b.subs(sols[0]);

#### and finally, x_2 must be....
f(b_1,b_2, b_3, b_4,b_5)= (A[(1),(0,1,2,3,4,)]*b())[0,0];
sols= solve( [f()== x_mu[1,0]] , b_2);
b = b.subs(sols[0]);
b = b.substitute(b_5==0)
```

The vector b is then

$$b = \begin{pmatrix} \frac{-2i\,\lambda\mu^2\rho u_1^4 - 4i\,\lambda^2\mu\rho u_1^3 + 2i\,h_1^2\lambda^2\mu u_1 - \left(-2i\,h_1^2\lambda\mu^2 + 2i\,\lambda^3\rho\right)u_1^2}{h_1^2\lambda p_\rho} \\ \frac{-i\,\lambda\mu^2\rho u_1^3 - 2i\,\lambda^2\mu\rho u_1^2 - i\,\lambda^3\rho u_1 + \left(i\,\lambda\mu^2 u_1 + 2i\,\lambda^2\mu + i\,\lambda u_1\right)h_1^2}{h_1^2\mu\rho u_1 + h_1^2\lambda\rho} \\ -\frac{\lambda\mu u_1^2 + \lambda^2 u_1}{h_1\lambda} \\ 0 \\ 0 \end{pmatrix}$$

In [32]:
```
b[0] = b[0].substitute( (-2*I*lambd*mu^2*rho*u_1^4 - 4*I*lambd^2*mu*rho*u_1^3 + 2*I*h_1^2*lambd^2*mu*u_1 - (-2*I*h_1^2*la
mbd*mu^2 + 2*I*lambd^3*rho)*u_1^2)/(h_1^2*lambd*p_rho) ==(-2*I*rho*u_1^2*(mu*u_1 +lambd)^3/(h_1^2*p_rho)  +2*I*mu*u_1*(mu
*u_1 +lambd)^2/p_rho));

num = numerator(b[1][0]);
num = num.expand();
num = num.collect(h_1);
num = num.substitute(-I*lambd*mu^2*rho*u_1^3 - 2*I*lambd^2*mu*rho*u_1^2 - I*lambd^3*rho*u_1 + (I*lambd*mu^2*u_1 + 2*I*lam
bd^2*mu + I*lambd*u_1)*h_1^2 ==  -I*lambd*rho*u_1(mu*u_1 + lambd)^2 + (I*lambd*mu^2*u_1 + 2*I*lambd^2*mu + I*lambd*u_1)*h
_1^2);

den = denominator(b[1][0]);
den = den.substitute(h_1^2*mu*rho*u_1 + h_1^2*lambd*rho == h_1^2*rho);
b[1] = num/den;
b[1] = (-I*(mu*u_1 + lambd)^2*lambd)/(h_1^2) + (I*lambd*mu^2*u_1 + 2*I*lambd^2*mu + I*lambd*u_1)/(rho);

b[2] = b[2].substitute((-(lambd*mu*u_1^2 + lambd^2*u_1)/(h_1*lambd)) ==(-u_1*(mu*u_1 + lambd)^2/h_1) )
```

```
/Applications/SageMath-8.3.app/Contents/Resources/sage/local/lib/python2.7/site-packages/IPython/core/interactives
hell.py:2882: DeprecationWarning: Substitution using function-call syntax and unnamed arguments is deprecated and
will be removed from a future release of Sage; you can use named arguments instead, like EXPR(x=..., y=...)
See http://trac.sagemath.org/5930 for details.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Which we can rewrite as

$$b = b(\mu) = \begin{pmatrix} -\frac{2i\,(\mu u_1+\lambda)^2\rho u_1^2}{h_1^2 p_\rho} + \frac{2i\,(\mu u_1+\lambda)\mu u_1}{p_\rho} \\ \frac{-i\,(\mu u_1+\lambda)^2\lambda\rho + \left(i\,\lambda\mu^2 u_1 + 2i\,\lambda^2\mu + i\,\lambda u_1\right)h_1^2}{(\mu u_1+\lambda)h_1^2\rho} \\ -\left(\frac{\mu u_1^2+\lambda u_1}{h_1}\right) \\ 0 \\ 0 \end{pmatrix}$$

The Lopatinski condition at $\lambda_0 = -u_1 + \sqrt{p_\rho} + \mathcal{O}(\epsilon)$

As we are looking for "decaying" manifolds we must have, for $x_1 > 0$, eigenspaces associated to the following eigenvalues:

$$\{\mu_1, \mu_2, \mu_3\} = \left\{-1 + \mathcal{O}(\epsilon^2), -1 + \mathcal{O}(\epsilon), \underbrace{-\frac{\lambda}{\alpha}}_{<0}\right\}$$

Analogously, "decaying" manifolds for $x_1 < 0$ must be eigenspaces associated to the following eigenvalues:

$$\mu_4 = \left\{+1 + \mathcal{O}(\epsilon^2)\right\}$$

The Lopatinski determinant Δ is of the form

$$\Delta = \Delta_0 + \Delta_1\epsilon + \Delta_2\epsilon^2 + \dots$$

so we need test only the lowest order term and show that it is not zero at the degenerated eigenvalue. As $\Delta_0 = \Delta_1 = 0$, we have

In [33]:
```
b= b.substitute(h_1=1/epsilon)
b_degenerated= b.substitute(rho = rho_plus, u_1= u_1_plus, p_rho=  p_rho_plus, xi=1) ;
b_degenerated=  b_degenerated.substitute(mu=x_0+epsilon*x_1+epsilon^2*x_2 + epsilon^3*x_3,lambd\
                                         = lambda_0 + epsilon*lambda_1+ epsilon^2*lambda_2 + epsilon^3*lambda_3);
b_degenerated = b_degenerated.expand();

#-----------------------------------------------------------------------------------
#Defining an auxiliary function that collects entries
#-----------------------------------------------------------------------------------
def get_coef(v,x,p,entry):
    #out = (v[entry][0]).coeff(x,p)     ##... after a few hours trying to figure out how to deal with indexing
    out = (v[entry][0]).coefficient(x,p)     ##... after a few hours trying to figure out how to deal with indexing
    return out

#-----------------------------------------------------------------------------------


b_zero = matrix([[get_coef(b_degenerated,epsilon,0,0)], [get_coef(b_degenerated,epsilon,0,1)], \
                 [get_coef(b_degenerated,epsilon,0,2)], [get_coef(b_degenerated,epsilon,0,3)], \
                 [get_coef(b_degenerated,epsilon,0,4)]]);
b_one = matrix([[get_coef(b_degenerated,epsilon,1,0)], [get_coef(b_degenerated,epsilon,1,1)],\
                [get_coef(b_degenerated,epsilon,1,2)], [get_coef(b_degenerated,epsilon,1,3)], \
                [get_coef(b_degenerated,epsilon,1,4)]]);
b_two = matrix([[get_coef(b_degenerated,epsilon,2,0)], [get_coef(b_degenerated,epsilon,2,1)], \
                [get_coef(b_degenerated,epsilon,2,2)], [get_coef(b_degenerated,epsilon,2,3)], \
                [get_coef(b_degenerated,epsilon,2,4)]]);

b_zero = b_zero.substitute(x_0 = -1, x_1 = 0 , x_2 = 0);
b_one  =  b_one.substitute(x_0 = -1, x_1 = 0 , x_2 = 0);
b_two  = b_two.substitute(x_0 = -1, x_1 = 0 , x_2 = 0);


Degenerated_Lopatinski = block_matrix(1,5,[X_1_one,b_zero,X_3_zero, X_4_one, X_5_zero]);
Degenerated_Lopatinski = Degenerated_Lopatinski(lambda_0 = (u_1_plus + sqrt(p_rho_plus)))

Degenerated_Lopatinski[0,1] = -2*I*u_1_plus;
Degenerated_Lopatinski[1,1] = -2*I*(u_1_plus + sqrt(p_rho_plus))*sqrt(p_rho_plus)/rho_plus;
Degenerated_Lopatinski[2,0] = -p_rho_plus;
Degenerated_Lopatinski[3,0] = i*u_1_plus*sqrt(p_rho_plus);
Degenerated_Lopatinski[4,0] = (u_1_plus + sqrt(p_rho_plus))*sqrt(p_rho_plus);
Degenerated_Lopatinski[0,4] = 2*u_1_plus*(rho_plus -rho_minus)/(u_1_plus - sqrt(p_rho_plus));
```

The output is

$$\Delta_2(degenerated) =$$

$$\begin{pmatrix}
0 & -2i\,u_{1^+} & 0 & 0 & -\frac{2\,(\rho^- - \rho^+)u_{1+}}{u_{1+}-\sqrt{p_{\rho^+}}} \\
0 & \frac{\left(-2i\,u_{1+}-2i\,\sqrt{p_{\rho^+}}\right)\sqrt{p_{\rho^+}}}{\rho^+} & 0 & 0 & \frac{\left(u_{1+}+\sqrt{p_{\rho^+}}\right)\rho^-}{\rho^+} - u_{1^+} - \sqrt{p_{\rho^+}} \\
-p_{\rho^+} & 0 & 0 & * & 0 \\
i\,\sqrt{p_{\rho^+}}u_{1^+} & 0 & -u_{1^+} - \sqrt{p_{\rho^+}} & i\left(u_{1^+} + \sqrt{p_{\rho^+}}\right)u_{1^-} + i\,u_{1^-}^2 & 0 \\
\left(u_{1^+} + \sqrt{p_{\rho^+}}\right)\sqrt{p_{\rho^+}} & 0 & i\,u_{1^+} & \frac{\left(u_{1+}+\sqrt{p_{\rho^+}}\right)^2 u_{1-}}{u_{1+}} + \frac{\left(u_{1+}+\sqrt{p_{\rho^+}}\right)u_{1-}^2}{u_{1+}} & 0
\end{pmatrix}$$

where

$$* = -\frac{\left(u_{1^+} + \sqrt{p_{\rho^+}}\right)^2 u_{1^-}}{u_{1^+}} - \frac{\left(u_{1^+} + \sqrt{p_{\rho^+}}\right)u_{1^-}^2}{u_{1^+}} - \left(u_{1^+} + \sqrt{p_{\rho^+}}\right)u_{1^-} - u_{1^-}^2$$

We just need to show that the determinant f this matrix is not vanishing. If we make use of the block structure of this matrix we reduce the problem to the evaluation of two determinants: one of a 2x2 matrix, other of a 3X3 matrix.

In [34]:
```
Degenerated_Lopatinski_2x2 = Degenerated_Lopatinski[(0,1),(1,4)];
Degenerated_Lopatinski_3x3 = Degenerated_Lopatinski[(2,3,4),(0,2,3)];
```

2x2 determinant:

$$\Delta_2(2x2) = \left(\begin{array}{c|c}
-2i\,u_{1^+} & -\frac{2\,(\rho^- - \rho^+)u_{1+}}{u_{1+}-\sqrt{p_{\rho^+}}} \\
\frac{\left(-2i\,u_{1+}-2i\,\sqrt{p_{\rho^+}}\right)\sqrt{p_{\rho^+}}}{\rho^+} & \frac{\left(u_{1+}+\sqrt{p_{\rho^+}}\right)\rho^-}{\rho^+} - u_{1^+} - \sqrt{p_{\rho^+}}
\end{array}\right)$$

which, using the Rankine Hugoniot condition, can be rewritten as

In [35]:
```
var(' R M');
Degenerated_Lopatinski_2x2 = Degenerated_Lopatinski_2x2.substitute(rho_plus ==R*rho_minus);
Degenerated_Lopatinski_2x2= Degenerated_Lopatinski_2x2.substitute(u_1_minus == R*u_1_plus);
Degenerated_Lopatinski_2x2= Degenerated_Lopatinski_2x2.substitute(p_rho_plus == (u_1_plus/M)^2);
Degenerated_Lopatinski_2x2= Degenerated_Lopatinski_2x2.substitute(p_rho_minus == u_1_plus^2/(M^2*R^(gamm-1)) );
Degenerated_Lopatinski_2x2 = Degenerated_Lopatinski_2x2.substitute(sqrt(u_1_plus^2/M^2) == u_1_plus/M);
Degenerated_Lopatinski_2x2 = Degenerated_Lopatinski_2x2.substitute(2*(R*rho_minus - rho_minus)*u_1_plus/\
                                                                   (u_1_plus - u_1_plus/M) ==2*rho_minus*(R-1)*M\
                                                                   /(M-1));
Degenerated_Lopatinski_2x2 = Degenerated_Lopatinski_2x2.substitute((-2*I*u_1_plus - 2*I*u_1_plus/M)*u_1_plus\
                                                                   /(M*R*rho_minus) == \
                                                                   -2*I*u_1_plus^2*(M+1)/(M^2*R*rho_minus));
Degenerated_Lopatinski_2x2 = Degenerated_Lopatinski_2x2.substitute(-u_1_plus + (u_1_plus + u_1_plus/M)/R\
                                                                   - u_1_plus/M  == u_1_plus*(M+1)*(1-R)/(M*R));
```

$$det\begin{pmatrix} -2i\,u_{1^+} & \frac{2\,(R-1)M\rho^-}{M-1} \\ -\frac{2i\,(M+1)u_{1^+}^2}{M^2R\rho^-} & -\frac{(R-1)(M+1)u_{1^+}}{MR} \end{pmatrix} = \frac{2i\,(R-1)(M+1)u_{1^+}^2}{MR} + \frac{4i\,(R-1)(M+1)u_{1^+}^2}{(M-1)MR} = \frac{2i\,(R-1)(M+1)^2u_{1^+}^2}{MR(M-1)} \neq 0 \tag{20}$$

In [36]:
```
detDegenerated_Lopatinski_2x2 = Degenerated_Lopatinski_2x2.determinant();
```

3x3 determinant: The idea here is analogous.

$$\Delta_2(3x3) = \begin{pmatrix} -p_{\rho^+} & 0 & -\frac{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)^2 u_{1^-}}{u_{1^+}} \backslash\text{-}\frac{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)u_{1^-}^2}{u_{1^+}} - \left(u_{1^+}+\sqrt{p_{\rho^+}}\right)u_{1^-} - u_{1^-}^2 \\ i\,\sqrt{p_{\rho^+}}\,u_{1^+} & -u_{1^+} - \sqrt{p_{\rho^+}} & i\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)u_{1^-} + i\,u_{1^-}^2 \\ \left(u_{1^+}+\sqrt{p_{\rho^+}}\right)\sqrt{p_{\rho^+}} & i\,u_{1^+} & \frac{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)^2 u_{1^-}}{u_{1^+}} + \frac{\left(u_{1^+}+\sqrt{p_{\rho^+}}\right)u_{1^-}^2}{u_{1^+}} \end{pmatrix}$$

In [37]:
```
var(' R M');
Degenerated_Lopatinski_3x3 = Degenerated_Lopatinski_3x3.substitute(rho_plus ==R*rho_minus);
Degenerated_Lopatinski_3x3 = Degenerated_Lopatinski_3x3.substitute(u_1_minus == R*u_1_plus);
Degenerated_Lopatinski_3x3 = Degenerated_Lopatinski_3x3.substitute(p_rho_plus == (u_1_plus/M)^2);
Degenerated_Lopatinski_3x3 = Degenerated_Lopatinski_3x3.substitute(p_rho_minus == u_1_plus^2/(M^2*R^(gamm-1)) );
Degenerated_Lopatinski_3x3 = Degenerated_Lopatinski_3x3.substitute(sqrt(u_1_plus^2/M^2) == u_1_plus/M);

Degenerated_Lopatinski_3x3 = Degenerated_Lopatinski_3x3.substitute((u_1_plus + u_1_plus/M) == u_1_plus*(M+1)/M);

Degenerated_Lopatinski_3x3 = Degenerated_Lopatinski_3x3.substitute((-2*I*u_1_plus - 2*I*u_1_plus/M)*u_1_plus\
                                                                   /(M*R*rho_minus) \
                                                                   == -2*I*u_1_plus^2*(M+1)/(M^2*R*rho_minus));
Degenerated_Lopatinski_3x3 = Degenerated_Lopatinski_3x3.substitute(-u_1_plus + (u_1_plus + u_1_plus/M)/R \
                                                                   - u_1_plus/M  == u_1_plus*(M+1)*(1-R)/(M*R));

aux = matrix([[1,0,0],[0,1,0],[0,i,1]])

Degenerated_Lopatinski_3x3  =       aux*Degenerated_Lopatinski_3x3;

Degenerated_Lopatinski_3x3 = Degenerated_Lopatinski_3x3.substitute((M + 1)*u_1_plus^2/M^2 \
                                                                   - u_1_plus^2/M  == u_1_plus^2/M^2 );

Degenerated_Lopatinski_3x3 = Degenerated_Lopatinski_3x3.substitute((M + 1)*R^2*u_1_plus^2/M \
                                                                   - R^2*u_1_plus^2 \
                                                                   + (M + 1)^2*R*u_1_plus^2/M^2 \
                                                                   - (M + 1)*R*u_1_plus^2/M == \
                                                                   R^2*u_1_plus^2/M + (M + 1)*R*u_1_plus^2/M^2);


aux = matrix([[1,0,0],[0,1,0],[0,i/M,1]])
Degenerated_Lopatinski_3x3  =       aux*Degenerated_Lopatinski_3x3;

Degenerated_Lopatinski_3x3[2,1] = -I*u_1_plus*(2*M +1)/M^2;
Degenerated_Lopatinski_3x3[2,2] =0;

aux = matrix([[1,-i/M,0],[0,1,0],[0,0,1]])
Degenerated_Lopatinski_3x3  =       aux*Degenerated_Lopatinski_3x3;
Degenerated_Lopatinski_3x3[0,2]  =-(2*M + 1)*R^2*u_1_plus^2/M - (M + 1)*R*u_1_plus^2*(2*M +1)/M^2 ;
```

$$
\begin{aligned}
det\begin{pmatrix} -\frac{u_{1+}^2}{M^2} & 0 & -\frac{(M+1)R^2u_{1+}^2}{M} - R^2u_{1^+}^2 - \frac{(M+1)^2Ru_{1+}^2}{M^2} - \frac{(M+1)Ru_{1+}^2}{M} \\ \frac{i\,u_{1+}^2}{M} & -u_{1^+} - \frac{u_{1+}}{M} & i\,R^2u_{1^+}^2 + \frac{i\,(M+1)Ru_{1+}^2}{M} \\ \frac{(M+1)u_{1+}^2}{M^2} & i\,u_{1^+} & \frac{(M+1)R^2u_{1+}^2}{M} + \frac{(M+1)^2Ru_{1+}^2}{M^2} \end{pmatrix} \\
= det\left(\begin{array}{ccc|c} -\frac{u_{1+}^2}{M^2} & 0 & & -\frac{(M+1)R^2u_{1+}^2}{M} - R^2u_{1^+}^2 - \frac{(M+1)^2Ru_{1+}^2}{M^2} - \frac{(M+1)Ru_{1+}^2}{M} \\ \frac{i\,u_{1+}^2}{M} & -u_{1^+} - \frac{u_{1+}}{M} & & i\,R^2u_{1^+}^2 + \frac{i\,(M+1)Ru_{1+}^2}{M} \\ \frac{u_{1+}^2}{M^2} & -\frac{i\,u_{1+}}{M} & & \frac{R^2u_{1+}^2}{M} + \frac{(M+1)Ru_{1+}^2}{M^2} \end{array}\right) \\
= det\begin{pmatrix} -\frac{u_{1+}^2}{M^2} & 0 & -\frac{(M+1)R^2u_{1+}^2}{M} - R^2u_{1^+}^2 - \frac{(M+1)^2Ru_{1+}^2}{M^2} - \frac{(M+1)Ru_{1+}^2}{M} \\ \frac{i\,u_{1+}^2}{M} & -u_{1^+} - \frac{u_{1+}}{M} & i\,R^2u_{1^+}^2 + \frac{i\,(M+1)Ru_{1+}^2}{M} \\ 0 & -\frac{i\,(2\,M+1)u_{1+}}{M^2} & 0 \end{pmatrix} \\
= det\begin{pmatrix} 0 & \frac{i\left(u_{1+}+\frac{u_{1+}}{M}\right)}{M} & -\frac{(2\,M+1)R^2u_{1+}^2}{M} - \frac{(M+1)(2\,M+1)Ru_{1+}^2}{M^2} \\ \frac{i\,u_{1+}^2}{M} & -u_{1^+} - \frac{u_{1+}}{M} & i\,R^2u_{1^+}^2 + \frac{i\,(M+1)Ru_{1+}^2}{M} \\ 0 & -\frac{i\,(2\,M+1)u_{1+}}{M^2} & 0 \end{pmatrix} \\
= -\frac{(2\,M+1)\left(\frac{(2\,M+1)R^2u_{1+}^2}{M} + \frac{(M+1)(2\,M+1)Ru_{1+}^2}{M^2}\right)u_{1^+}^3}{M^3} = -\frac{(2\,M+1)^2Ru_{1^+}^5(RM+M+1)}{M^5} \neq 0
\end{aligned}
\tag{21}
$$

In [38]:
```
detDegenerated_Lopatinski_3x3 = Degenerated_Lopatinski_3x3.determinant();
```

Combining the results in (20) and (21) we get that $\Delta_2 \neq 0$, therefore this is not a root of the Lopatinski determinant.

---

[1] I.e., the zeroth- order terms in ϵ.

---

[2] \footnote{Hersh's lemma (\cite{Hersh}) consists in proving that the matrix $-A_1^{-1}(\lambda A_0 + i\xi A_2)$ never has a purely imaginary eigenvalue. In fact, if that is the case and $i\mu$, $\mu \in \mathbb{R}$, is a purely imaginary eigenvalue then $det(-A_1^{-1}(\lambda A_0 + i\xi A_2) - i\mu) = 0$ and then that $det(\frac{\lambda}{i}A_0 + \xi A_2 + \mu A_1) = 0$. As this operator is hyperbolic (as shown by Blake in one of his worklogs and in section ???), it implies that $\frac{\lambda}{i} \in \mathbb{R}$, which can only be the case if λ is purely imaginary. However, this is not in the domain of solutions we allow thourgh Laplace transform. Consequently, since the eigenvalues of This contradiction shows that since $-A_1^{-1}(\lambda A_0 + i\xi A_2)$ depend continuously on λ and ξ the number of positive and negative eigenvalues remains the same as (λ, ξ) vary (and $i\lambda \notin \mathbb{R}$). It suffices then to analyse the number of positive and negative eigenvalues of $-A_1^{-1}A_0$.

[3] Essentially, we use the fact that $X_1^{(0)} = X_2^{(0)} = X_2^{(1)} = X_3^{(1)} = X_4^{(0)} = X_5^{(1)} = X_5^{(2)} = 0$ (see this section).

[4]We will mainly use the fact that $det(X_1^{(1)}, X_2^{(2)}, X_3^{(0)}, X_4^{(2)}, X_5^{(1)}) = -det(X_2^{(2)}, X_4^{(2)}, X_1^{(1)}, X_3^{(0)}, X_5^{(1)})$

In [ ]:
```
```