# Oregonator model for BZ reaction

by

Jongyoon Sohn

Final Project in MATH 551

Department of Statistics, Eberly College of Science

Pennsylvania State University

December 8, 2017

## Abstract:

BZ reaction is a chemical non-equilibrium reaction with respect to time. A mathematical model Oregonator, developed by Richard Field and Richard M. Noyes at University of Oregon in 1974, simulates the BZ reaction. The model consists of a system of stiff ODEs and numerical errors are obtained by using ode23s and the Forward Euler method. Stability states are investigated in terms of the trace and determinant of the Jacobian matrix.

## 1. Introduction

Boris Belousov was a scientist in chemistry born in Russia. He studied the Kreb's cycle model which led him to the discovery of a new oscillating reaction. Anatol Zhabotinsky, a Russian biophysicist, rediscovered Belousov's work in graduate school in the 60's. He published Belousov's findings in 1981 and received the Lenin prize for their discovery, the highest soviet chemistry prize.

A BZ reaction is known as an example of non-equilibrium thermodynamics. These reactions are far from equilibrium and remain so for a significant length of time and evolve chaotically.

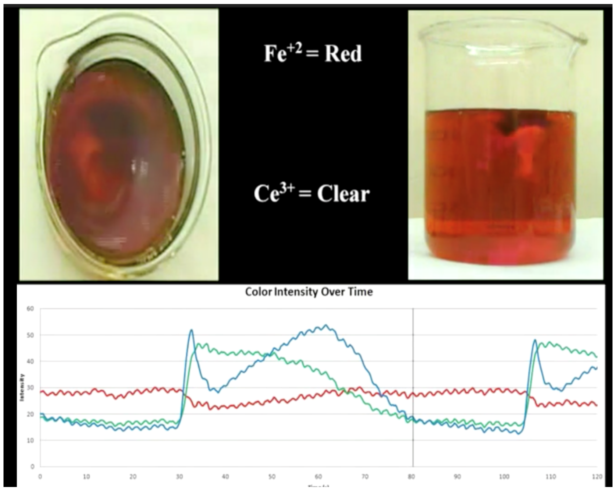


*Figure 1 Change in color*

Colors represent products of chemical reactions and the change of colors with respect to time shows that reactions are not in an equilibrium state. The second law of thermodynamic states that the reaction must move towards an irreversible state of equilibrium, increasing randomness or disorders. This reactions are of nonlinear thermodynamics. Even though it doesn't seem for a short period of time they go to equilibrium eventually, which means that they do not violate the second law of thermodynamics.

## 2. Numerical analysis in the Oregonator model

The three-dimensional Oregonator model consists of kinetic equations,

$$\begin{cases} \frac{dX}{dt} = k_1H^2AY - k_2HXY + k_3HAX - 2k_4X^2 \\ \frac{dY}{dt} = -k_1H^2AY - k_2HXY + hk_cBZ \\ \frac{dZ}{dt} = 2k_3HAX - k_cBZ \end{cases} \qquad (1)$$

where $A = [BrO_3^-], B = [CH_2(COOH)_2], H = [H^+], X = [BrO_2], Y = [Br^-], Z = [Ce^{4+}], stoichiometric\ coefficient\ h.$

According to Marco (1995), it is known that the equation has a unique steady state with positive values for all values of the parameters but the stability of the steady state depends on the concentrations A and B , kinetic parameters, and stoichiometric coefficient h. When the steady state is unstable the model will behave in oscillation.

The system of kinetic equations can be modified in a dimensionless form.

$$\begin{cases} \epsilon\frac{dx}{d\tau} = qay - xy + x(a - x) \\ \epsilon'\frac{dy}{d\tau} = -qay - xy + fbz \\ \frac{dz}{d\tau} = ax - bz \end{cases} \qquad (2)$$

where $x = \frac{X}{X_0}, X_0 = \frac{k_3HA_0}{2k_4}, y = \frac{Y}{Y_0}, Y_0 = \frac{k_3A_0}{k_2}, z = \frac{Z}{Z_0}, Z_0 = \frac{(k_3HA)^2}{k_ck_4B_0}, \tau = \frac{t}{t_0}, t_0 = \frac{1}{k_cB_0}, \epsilon = \frac{k_cB_0}{k_3HA_0} = 9.9 * 10^{-3}, \epsilon' = \frac{2k_ck_4B}{k_2k_3A} = 1.98 * 10^{-5}, q = \frac{2k_1k_4}{k_2k_3} = 7.62 * 10^{-5}$

The following is the plot of the system of ODEs and some parameters, a, f and b, are set to be 1 with the initial condition of x(0)=.0005, y(0)=.02, z(0)=.02. Since the model consists of a system of stiff IVPs and there is no analytic solution that can be found, the ode23s function in Matlab was used to solve a system of stiff equations. The result from ode23s is considered as an analytical solution.

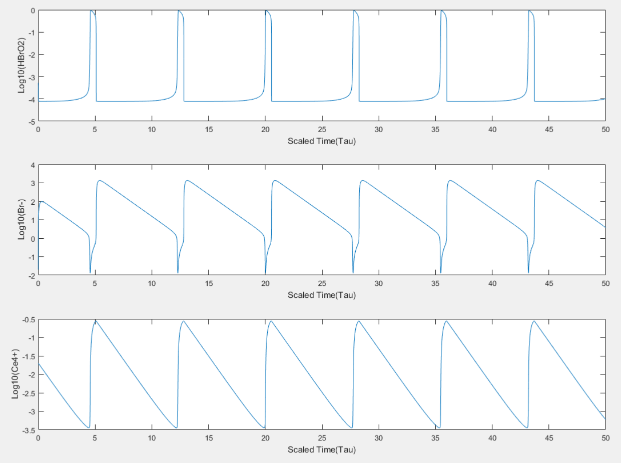


Figure 2 Concentration of x, y, z for t = 0 to 50

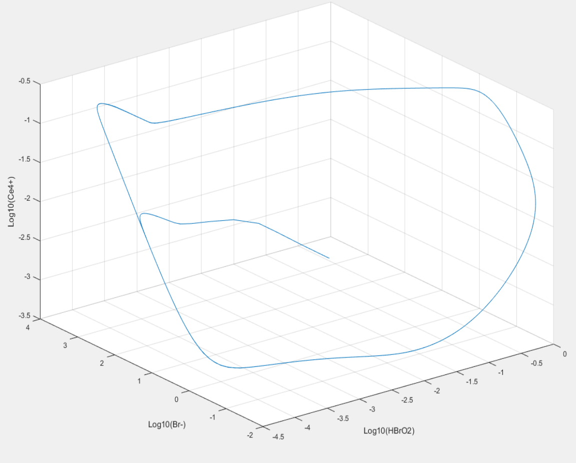


Figure 3 Attractor for Oregonator model

The plots show that the model is oscillating consistently for the length of time of 50. The ode45 function in Matlab was also used to plot for the same figures and they look alike. In this sense, ode23s is preferred to produce plots of a stiff ODE with less time taken.

## 3. Error analysis

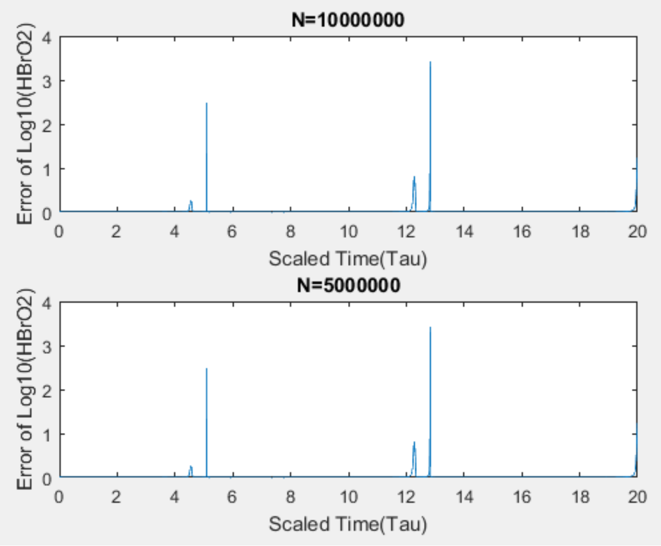
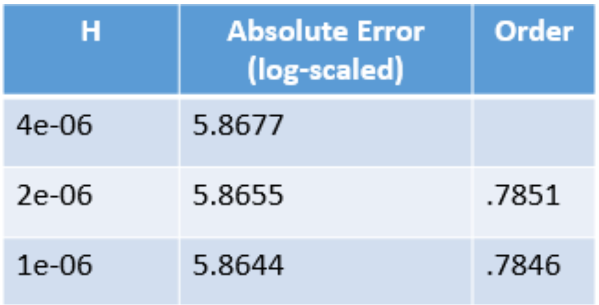


Figure 4 Absolute error of logarithm



| H | Absolute Error (log-scaled) | Order |
|---|---|---|
| 4e-06 | 5.8677 | |
| 2e-06 | 5.8655 | .7851 |
| 1e-06 | 5.8644 | .7846 |

Figure 5 A plot of step size with corresponding absolute error

I used the Forward Euler method to analyze the numerical errors compared to an adaptive method, ode23s. The length steps are necessary for the FE method to work well for a stiff ODE because the stability for the method is dependent on the step size. In Figure 4, the variable x is considered only. Even the number of steps of $10^7$ does not really produce a good approximation compared to the one from ode23s, producing its absolute error of 5.8677. In Figure 5, the numerical order of FE method is .78, slightly less than the analytical order of 1.

## 4. Analysis in stability

By letting each equation in (a) equal to zero, a simpler system of ODEs is obtained below.

$$\begin{cases} \epsilon \dfrac{dx}{d\tau} = ax - x^2 - fbz\dfrac{x - qa}{x + qa} \\ \dfrac{dz}{d\tau} = ax - bz \end{cases} \qquad (3)$$

As stated earlier, kinetic parameters a and b and a stoichiometric coefficient h affect the region of the stable steady state.
A positive steady state of x can be calculated

$$x_m = \frac{1 - q - f + \sqrt{(1 - q - f)^2 + 4q(f + 1)}}{2}$$

and the stability for the positive steady state will be analyzed by plugging $x_m$ into (2) again.

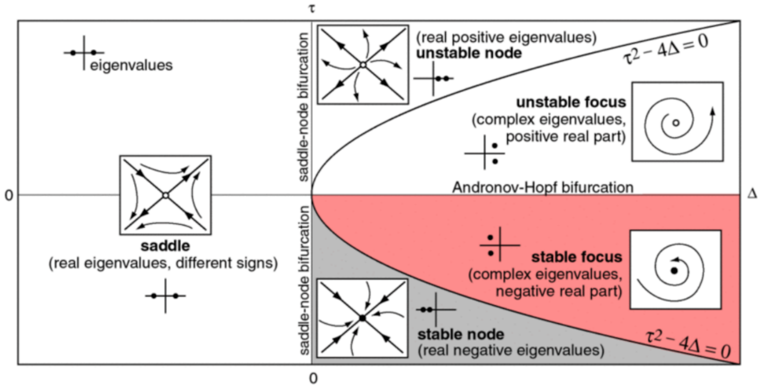


*Figure 4 Stability, Bifurcation*

The trace and the determinant of the Jacobian matrix have been obtained.

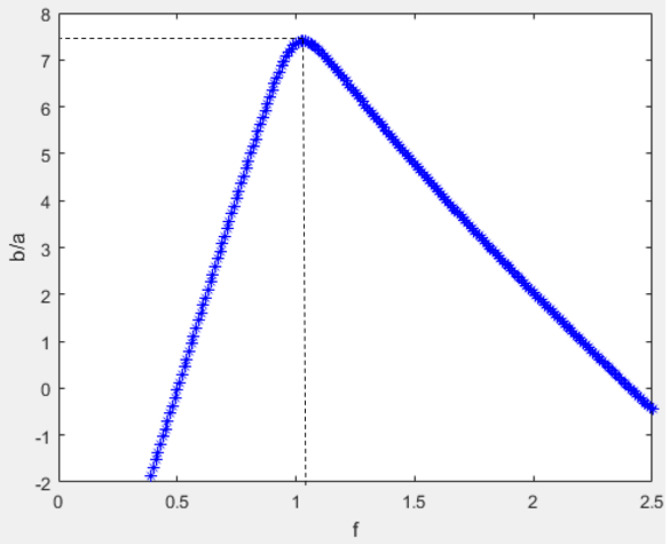
$$trace(J) = \frac{a}{\epsilon}\left(1 - 2x_m - \frac{2qfx_m}{(x_m + q)^2}\right) - b$$

$$det(J) = \frac{ab(x_m^2 + qx_m + 2qy_m)}{\epsilon(x_m + q)} > 0$$

The determinant of the Jacobian matrix from given parameters is always greater than zero so the stable steady states occur if the trace is less than zero. Setting the trace equal to zero, we get

$$\frac{b}{a} = \frac{1}{\epsilon}\left(1 - 2x_m - \frac{2qfx_m}{(x_m + q)^2}\right)$$

The relationship between the ratio of kinetic parameters and a stoichiometric coefficient is shown below.



Inside the curve, the trace of the Jacobian matrix is greater than zero so the steady state when parameters lie inside the curve is unstable. On the other hand, the steady state when parameters lie outside the curve is stable because the trace gets less than zero. Bifurcation occurs when an equilibrium solution appears or disappears or splits into two more equilibrium solutions. According to Figure 6, it occurs when the trace is zero given the determinant greater than zero. Therefore, Hopf bifurcation happends when the parameters lie on the curve.

## 5. Conclusion

The goal of this project is to analyze numerical errors by using appropriate methods. For a stiff IVPs, it is proved that the Forward Euler is not a good method to use because of its properties of conditional stability. The Forward Euler method took tremendous time to produce an output that looks close to the one from odes23s. Since we want to see results in a less time with a higher accuracy, the Backward Euler or adaptive numerical methods are recommended to produce desired outputs for a stiff ODE.

# References:

Judson, T. W. (n.d.). The Ordinary Differential Equations Project. Retrieved from http://faculty.sfasu.edu/judsontw/ode/html/firstlook07.html

Belousov-Zhabotinsky The Reaction Documentary. (n.d.). Retrieved from https://www.youtube.com/watch?v=nEncoHs6ads

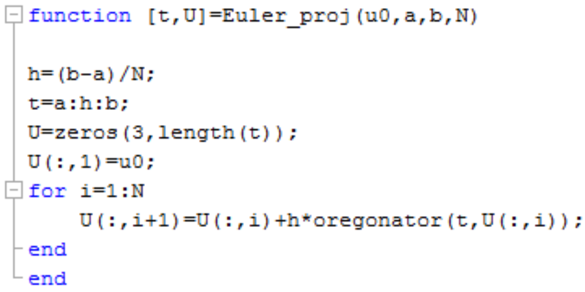
(n.d.). Retrieved from https://en.wikipedia.org/wiki/Laws_of_thermodynamics#Second_law

(n.d.). Retrieved from http://www.scholarpedia.org/article/Equilibrium

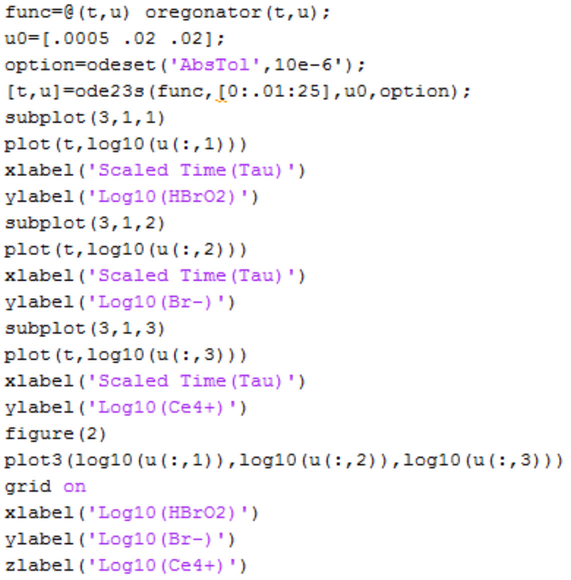
(n.d.). Retrieved from http://www.scholarpedia.org/article/Oregonator

Mazzotti, M., Morbidelli, M., & Serravalle, G. (1995). Bifurcation Analysis of the Oregonator Model in the 3-D Space BromateMalonic Acid/Stoichiometric Coefficient . *Journal of Physical Chemistry,99*(13), 4501-4511. doi:10.1021/j100013a020

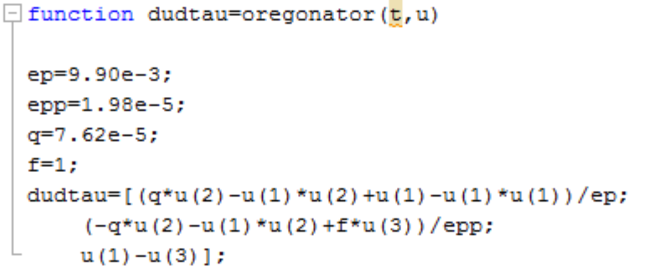
# Appendix:



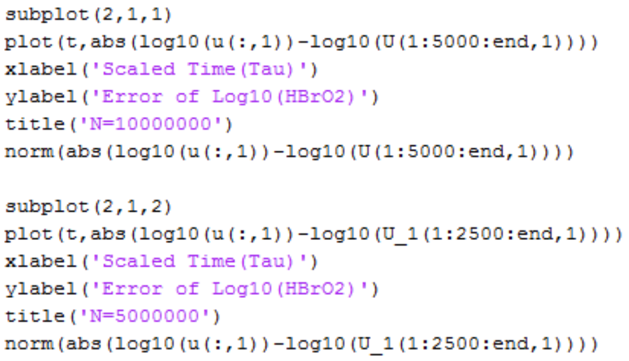
```
function [t,U]=Euler_proj(u0,a,b,N)

h=(b-a)/N;
t=a:h:b;
U=zeros(3,length(t));
U(:,1)=u0;
for i=1:N
    U(:,i+1)=U(:,i)+h*oregonator(t,U(:,i));
end
end
```



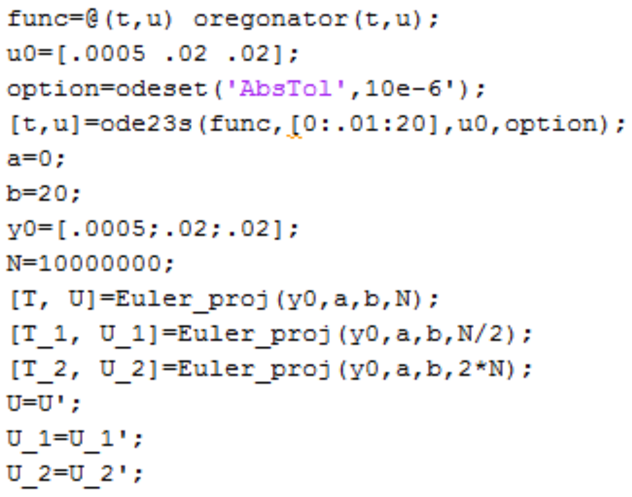
```
func=@(t,u) oregonator(t,u);
u0=[.0005 .02 .02];
option=odeset('AbsTol',10e-6');
[t,u]=ode23s(func,[0:.01:25],u0,option);
subplot(3,1,1)
plot(t,log10(u(:,1)))
xlabel('Scaled Time(Tau)')
ylabel('Log10(HBrO2)')
subplot(3,1,2)
plot(t,log10(u(:,2)))
xlabel('Scaled Time(Tau)')
ylabel('Log10(Br-)')
subplot(3,1,3)
plot(t,log10(u(:,3)))
xlabel('Scaled Time(Tau)')
ylabel('Log10(Ce4+)')
figure(2)
plot3(log10(u(:,1)),log10(u(:,2)),log10(u(:,3)))
grid on
xlabel('Log10(HBrO2)')
ylabel('Log10(Br-)')
zlabel('Log10(Ce4+)')
```



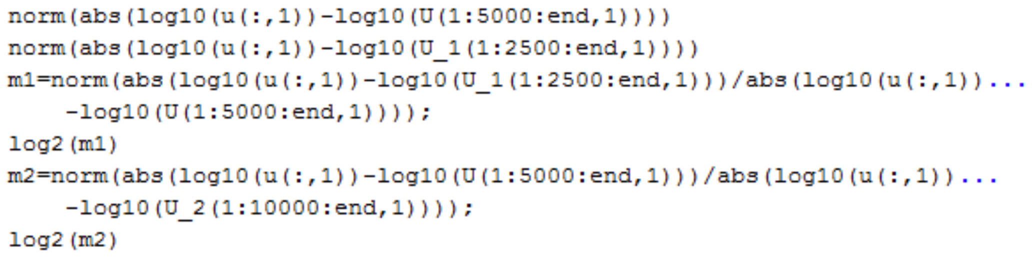
```
function dudtau=oregonator(t,u)

ep=9.90e-3;
epp=1.98e-5;
q=7.62e-5;
f=1;
dudtau=[(q*u(2)-u(1)*u(2)+u(1)-u(1)*u(1))/ep;
    (-q*u(2)-u(1)*u(2)+f*u(3))/epp;
    u(1)-u(3)];
```



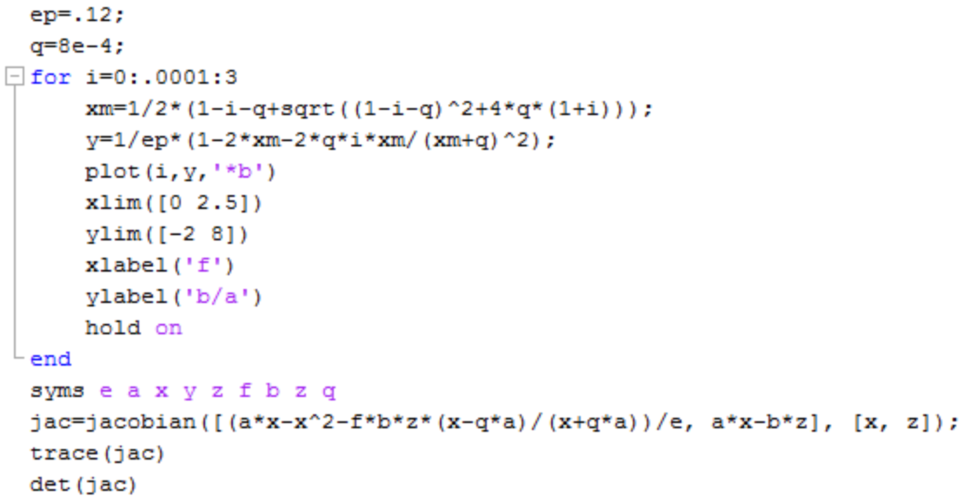
```
subplot(2,1,1)
plot(t,abs(log10(u(:,1))-log10(U(1:5000:end,1))))
xlabel('Scaled Time(Tau)')
ylabel('Error of Log10(HBrO2)')
title('N=10000000')
norm(abs(log10(u(:,1))-log10(U(1:5000:end,1))))

subplot(2,1,2)
plot(t,abs(log10(u(:,1))-log10(U_1(1:2500:end,1))))
xlabel('Scaled Time(Tau)')
ylabel('Error of Log10(HBrO2)')
title('N=5000000')
norm(abs(log10(u(:,1))-log10(U_1(1:2500:end,1))))
```



```
func=@(t,u) oregonator(t,u);
u0=[.0005 .02 .02];
option=odeset('AbsTol',10e-6');
[t,u]=ode23s(func,[0:.01:20],u0,option);
a=0;
b=20;
y0=[.0005;.02;.02];
N=10000000;
[T, U]=Euler_proj(y0,a,b,N);
[T_1, U_1]=Euler_proj(y0,a,b,N/2);
[T_2, U_2]=Euler_proj(y0,a,b,2*N);
U=U';
U_1=U_1';
U_2=U_2';
```



```
norm(abs(log10(u(:,1))-log10(U(1:5000:end,1))))
norm(abs(log10(u(:,1))-log10(U_1(1:2500:end,1))))
m1=norm(abs(log10(u(:,1))-log10(U_1(1:2500:end,1))))/abs(log10(u(:,1))...
    -log10(U(1:5000:end,1))));
log2(m1)
m2=norm(abs(log10(u(:,1))-log10(U(1:5000:end,1))))/abs(log10(u(:,1))...
    -log10(U_2(1:10000:end,1))));
log2(m2)
```



```
ep=.12;
q=8e-4;
for i=0:.0001:3
    xm=1/2*(1-i-q+sqrt((1-i-q)^2+4*q*(1+i)));
    y=1/ep*(1-2*xm-2*q*i*xm/(xm+q)^2);
    plot(i,y,'*b')
    xlim([0 2.5])
    ylim([-2 8])
    xlabel('f')
    ylabel('b/a')
    hold on
end
syms e a x y z f b z q
jac=jacobian([(a*x-x^2-f*b*z*(x-q*a)/(x+q*a))/e, a*x-b*z], [x, z]);
trace(jac)
det(jac)
```