姓名：张楚珩

老师：李文飞

计算物理

2014年4月17日

Lorenz Model

一、算法流程与计算公式：

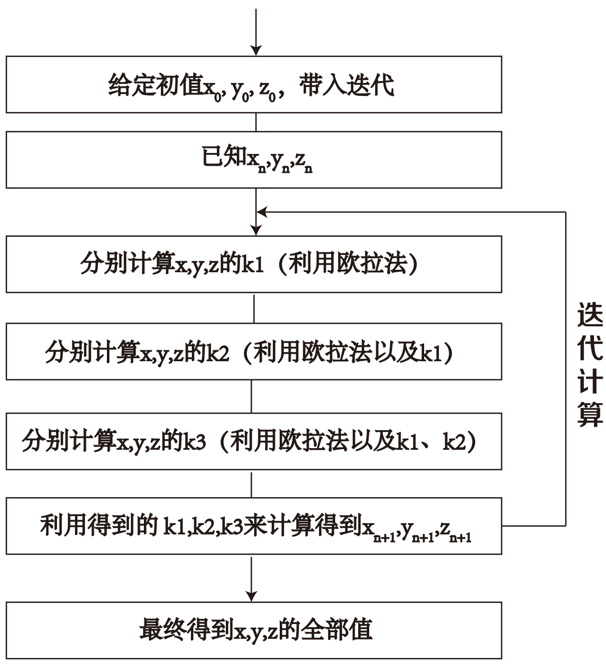
利用了四阶Runge-Kutta方法，根据现已推演到的值x_n,y_n,z_n和已知的微分方程，利用公式

$$y_{n+1} = y_n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$\begin{cases} k_1 = & f(y_n, t_n) \\ k_2 = & f(y_n + k_1 \Delta t/2, t_{n+1/2}) \\ k_3 = & f(y_n + k_2 \Delta t/2, t_{n+1/2}) \\ k_4 = & f(y_n + k_3 \Delta t, t_{n+1}) \end{cases}$$

来进行计算。

其计算的流程图如下：



二、程序源代码（C语言）：

```
//
//  main.c
//  ComputationalPhysics
//
//  Created by ZHANG CH on 14-4-16.
//  Copyright (c) 2014年 NJU. All rights reserved.
//

//PROBLEM STATING:
//dx/dt = 10(y-x)
//dy/dt = -xz + rx - y
//dz/dt = xy - (8/3)z
//  PRO 1:
//  r = 25 Plot z-x plane
//  PRO 2:
//  change the val of r, plot z-x plane
//
//  1. Draw flow diagram
//  2. Make program
//  3. Give out computation results and discuss the results

//Formula implemented:
//{x_n+1} = {x_n} + (DELTA_T / 6) * (k1 + 2 k2 + 2 k3 + k4)
//k1 = f({x_n})
//k2 = f({x_n + k1*(DELTA_T/2)})
//k2 = f({x_n + k2*(DELTA_T/2)})
//k4 = f({x_n + k3*(DELTA_T)})

//input data:   25  25  0   100    \\  12    2   9

#include <stdio.h>
#define DELTA_T (0.001)
#define NUM_OF_POINTS (30010)
#define SIGMA (10.0)
#define B (8.0/3.0)

double x[NUM_OF_POINTS] = {0};
double y[NUM_OF_POINTS] = {0};
double z[NUM_OF_POINTS] = {0};
int numOfLoop;
double r_min, r_max, r_step, s;
double x_ini,y_ini,z_ini;

FILE * fp;

void init();
void run();
void print(double * outputSet);
void calNext(int i, double r);
double fun(double x, double y, double z, double r, char label);

int main(int argc, const char * argv[])
{
```

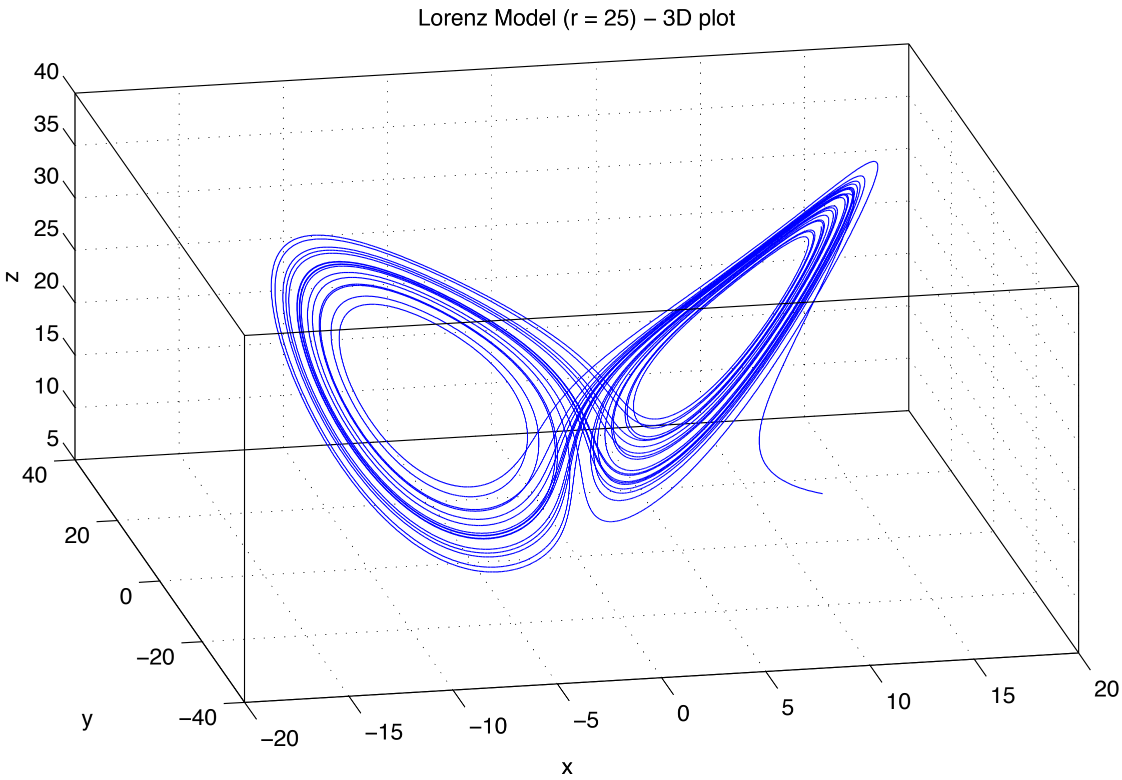
```
    init();
    //You can uncomment below and comment init() function to auto operate the
program
    //r_min = 25; r_max = 25; r_step = 0; s = 30000;
    //x_ini = 12; y_ini = 2; z_ini = 9;
    run();
    return 0;
}

void init()
{
    printf("Please input the value of r (r_min, r_max, step) and steps to in-
volve s:\n");
    scanf("%lf%lf%lf%lf",&r_min, &r_max, &r_step, &s);
    printf("Please input the initial value of x,y,z:\n");
    scanf("%lf%lf%lf",&x_ini, &y_ini, &z_ini);
}

void run()
{
    double r = r_min;
    int i,j;
    numOfLoop = (r_min == r_max) ? 1 : (int)((r_max - r_min) / r_step);
    fp = fopen("output.txt", "wb");
    for (j=0; j<numOfLoop; j++)
    {
        x[0] = x_ini;
        y[0] = y_ini;
        z[0] = z_ini;

        for (i=0; i<s; i++)
        {
            calNext(i, r);
        }

        fprintf(fp, "The data below is array of x:\n");
        print((double *)x);
        fprintf(fp, "The data below is array of y:\n");
        print((double *)y);
        fprintf(fp, "The data below is array of z:\n");
        print((double *)z);

        r += r_step;
    }
    fclose(fp);
}

void print(double * outputSet)
{
    int i;
    fprintf(fp, "[");
    for (i=0; i<s; i++)
    {
        if (i != 0) fprintf(fp, ",");
        fprintf(fp, "%lf ", outputSet[i]);
    }
    fprintf(fp, "]\n\n");
}
```

```
void calNext(int i, double r)
{
    double k1x = fun(x[i], y[i], z[i], r, 'x');
    double k1y = fun(x[i], y[i], z[i], r, 'y');
    double k1z = fun(x[i], y[i], z[i], r, 'z');

    double k2x = fun(x[i] + k1x*(DELTA_T/2.0), y[i] + k1y*(DELTA_T/2.0), z[i]
+ k1z*(DELTA_T/2.0), r, 'x');
    double k2y = fun(x[i] + k1x*(DELTA_T/2.0), y[i] + k1y*(DELTA_T/2.0), z[i]
+ k1z*(DELTA_T/2.0), r, 'y');
    double k2z = fun(x[i] + k1x*(DELTA_T/2.0), y[i] + k1y*(DELTA_T/2.0), z[i]
+ k1z*(DELTA_T/2.0), r, 'z');

    double k3x = fun(x[i] + k2x*(DELTA_T/2.0), y[i] + k2y*(DELTA_T/2.0), z[i]
+ k2z*(DELTA_T/2.0), r, 'x');
    double k3y = fun(x[i] + k2x*(DELTA_T/2.0), y[i] + k2y*(DELTA_T/2.0), z[i]
+ k2z*(DELTA_T/2.0), r, 'y');
    double k3z = fun(x[i] + k2x*(DELTA_T/2.0), y[i] + k2y*(DELTA_T/2.0), z[i]
+ k2z*(DELTA_T/2.0), r, 'z');

    double k4x = fun(x[i] + k3x*(DELTA_T), y[i] + k3y*(DELTA_T), z[i] +
k3z*(DELTA_T), r, 'x');
    double k4y = fun(x[i] + k3x*(DELTA_T), y[i] + k3y*(DELTA_T), z[i] +
k3z*(DELTA_T), r, 'y');
    double k4z = fun(x[i] + k3x*(DELTA_T), y[i] + k3y*(DELTA_T), z[i] +
k3z*(DELTA_T), r, 'z');

    x[i+1] = x[i] + (DELTA_T / 6.0) * (k1x + 2*k2x + 2*k3x + k4x);
    y[i+1] = y[i] + (DELTA_T / 6.0) * (k1y + 2*k2y + 2*k3y + k4y);
    z[i+1] = z[i] + (DELTA_T / 6.0) * (k1z + 2*k2z + 2*k3z + k4z);
}

double fun(double x, double y, double z, double r, char label)
{
    switch (label) {
        case 'x':
            return (SIGMA * (y - x));
            break;

        case 'y':
            return ((r*x) - (x*z) - y);
            break;

        case 'z':
            return ((x*y) - (B*z));
            break;

        default:
            return 0;
            break;
    }
}
```

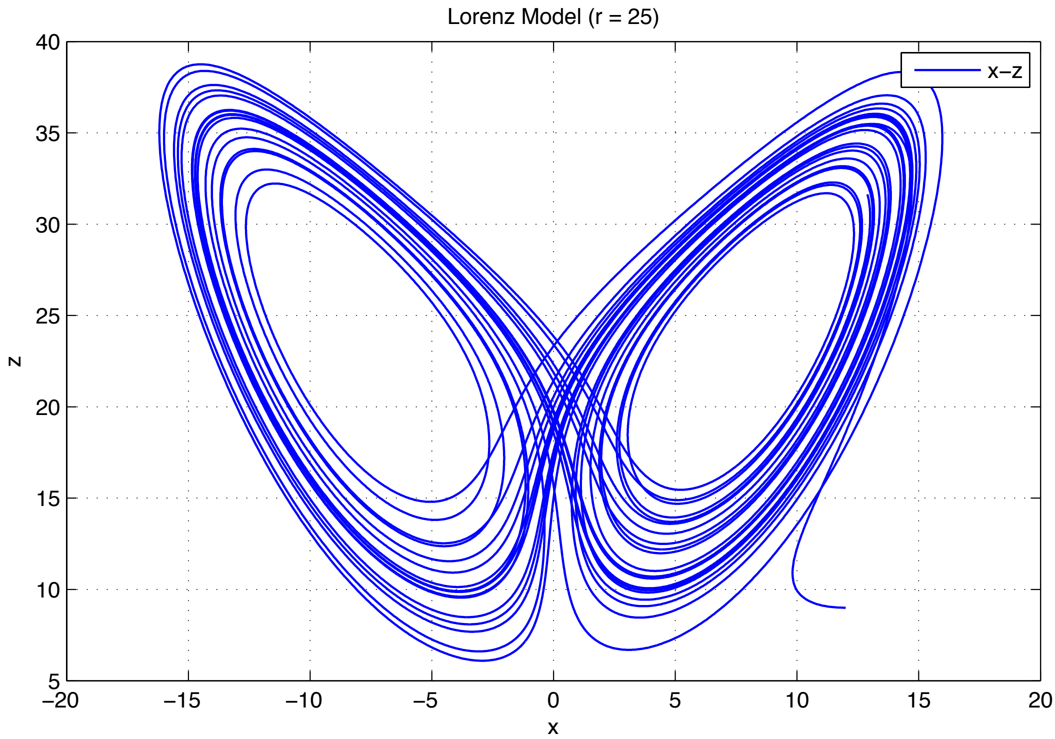
三、计算结果:

选定初值 x(0) = 12, y(0) = 2, z(0) = 9，时间t从0到30，时间步长取0.01。

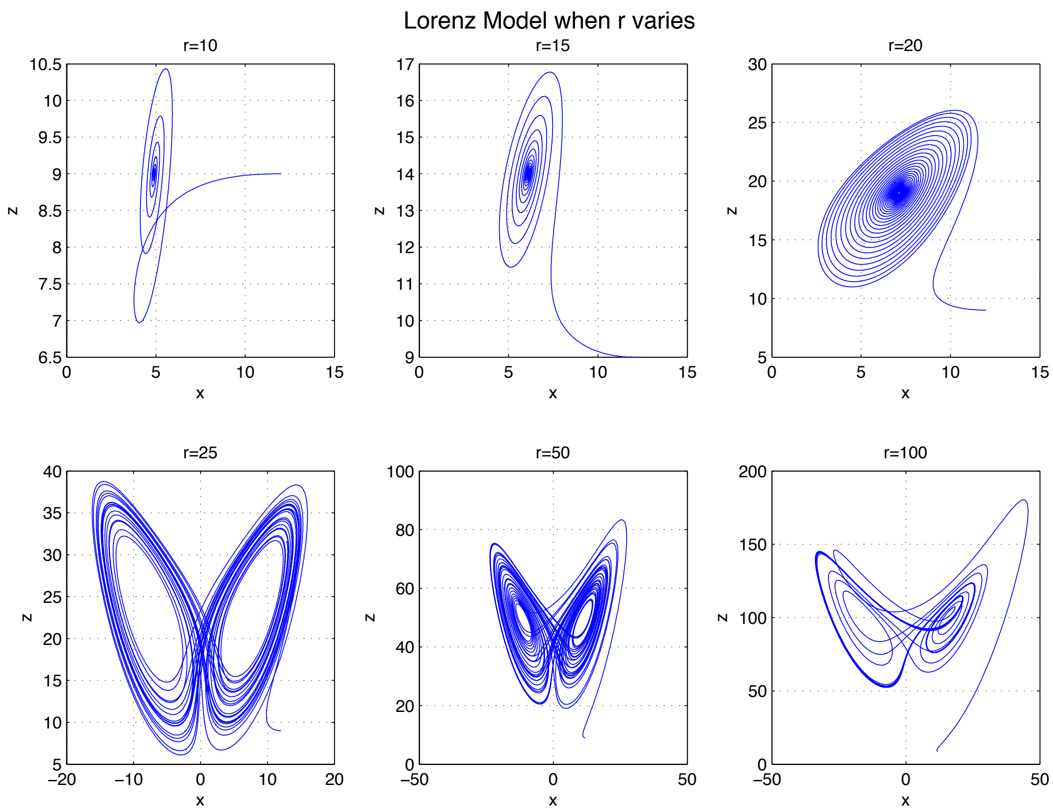
可得到r=20时的3D图形：



把图形向x-z平面上投影，可以得到x-z的关系图：



之后，改变r的值，可以得到不同r值下的x-z平面连线图：



我们看到，当r逐渐增大的时候，x-z的图像由x > 0的趋于稳定的构型，转化为了一个x可正可负的混沌构型。

为了找到构型发生突变处的r值，我们在上一组数据中发生突变的r=20和r=25之间又插入了一系列的值，得到了以下图形：

这样我们可以找到，发生突变的地方在r=20 与 r=21之间。