**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ**

**имени академика С.П.КОРОЛЕВА (национальный исследовательский университет)»**

*ФАКУЛЬТЕТ ИНФОРМАТИКИ*

*КАФЕДРА ТЕХНИЧЕСКОЙ КИБЕРНЕТИКИ*

СТАТИСТИЧЕСКИЙ АНАЛИЗ И МОДЕЛИРОВАНИЕ

ПРОЦЕССОВ АВТОРЕГРЕССИИ И СКОЛЬЗЯЩЕГО СРЕДНЕГО

курсовая работа по дисциплине «Теория случайных процессов»

Вариант № 56

*Выполнил:* Витальев А.В.

*Группа:* 6309

*№ зачётной книжки:* 136155

*Проверил:* Храмов А.Г.

*Оценка:* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Дата:* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Самара 2016

**Задание**

1. Изобразить графически фрагмент исходного случайного процесса (СП). Оценить моментные функции (МФ) исходного, рассчитав выборочные среднее, дисперсию и нормированную корреляционную функцию (НКФ). Оценить интервал корреляции СП. Изобразить графически оценку НКФ исходного СП.

2. Построить модели авторегрессии АР(M) = АРСС(M, 0) порядков M = 1, 2, 3 (всего 3 модели) на основе решения системы уравнений Юла–Уокера. Для каждой модели рассчитать теоретические НКФ выходной последовательности. На основе сравнения выборочной НКФ и теоретических НКФ выбрать наилучшую модель СП в классе моделей АР.

3. Построить модели скользящего среднего СС(N) = АРСС(0, N) порядков N = 0, 1, 2, 3 (всего 4 модели) на основе решения системы нелинейных уравнений. Для каждой модели рассчитать теоретические НКФ выходной последовательности. На основе сравнения выборочной НКФ и теоретических НКФ выбрать наилучшую модель СП в классе моделей СС.

4. Построить смешанные модели авторегрессии – скользящего среднего (АРСС(M, N) до третьего порядка включительно (M = 1, 2, 3; N = 1, 2, 3) (всего 9 моделей). Рассчитать теоретические НКФ выходной последовательности для различных порядков моделей АРСС. На основе сравнения выборочной и теоретических НКФ выбрать наилучшую модель СП в классе смешанных моделей.

5. Для каждой из трёх лучших моделей (АР, СС, АРСС) записать системы уравнений для расчёта параметров модели, записать системы уравнений для расчёта теоретической КФ, смоделировать СП, рассчитать выборочные МФ, сравнить их с выборочными МФ исходного СП и с теоретическими МФ. Для каждой из этих трёх моделей сравнить графически НКФ: (1) выборочную исходного СП, (2) теоретическую, (3) выборочную смоделированного СП.

6. Изготовить таблицу сравнения МФ и расчёта качества для трёх лучших моделей. Изобразить графически фрагмент реализации СП, сгенерированного по наилучшей модели.

**Исходные данные**

Число отсчётов в выборке: 5000

Выборка:

-10.854 -16.747 -15.147 -12.109 -13.816 -18.653 -20.898 -14.015 -14.489 -10.088

-20.431 -7.679 -13.950 -21.770 -13.079 -15.257 -9.675 -5.034 -15.720 -5.565

...

-5.800 -12.680 -15.243 -8.778 -19.417 -14.739 -18.472 -12.669 -16.699 -6.514

# Моментные функции случайного процесса

Пусть дана выборка  из  отсчётов стационарного в широком смысле эргодического дискретного случайного процесса .

Изобразим графически исходный случайный процесс.

*Рисунок 1 – Графическое представление исходного процесса.*

Оценим моментные функции этого процесса.

Выборочное среднее – оценка математического ожидания  – рассчитывается по формуле: , где  – компоненты вектора .

Получаем .

Формула для расчёта выборочной дисперсии имеет вид:  
,  
но мы воспользуемся исправленной дисперсией, рассчитывающейся по формуле  
, так как она является лучшей (несмещённой) оценкой дисперсии процесса.

Получаем: .

Формула для оценки корреляционной функции :  
.  
Это исправленная выборочная корреляционная функция.

В таблицу 1 запишем 11 первых значений этой функции. Заметим, что  что и следовало ожидать.

Оценим нормированную корреляционную функцию, которая поможет количественно оценить корреляцию сечений:  
.

*Таблица 1 –Значения выборочных корреляционных функций*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
|  | 35.4516 | -9.3586 | -2.3908 | 6.2819 | -3.5018 | -0.4541 |
|  | 1.0000 | -0.2639 | -0.0674 | 0.1771 | -0.0987 | -0.0128 |
|  | 6 | 7 | 8 | 9 | 10 |  |
|  | 2.2937 | -1.6298 | 0.2607 | 1.2295 | -0.5354 |  |
|  | 0.0647 | -0.0459 | 0.0073 | 0.0346 | -0.0151 |  |

Оценим радиус корреляции случайного процесса  по формуле:  


только вместо  используем , а  будем полагать не слишком большим по сравнению с размером выборки во избежание чрезмерной ошибки:

.

Получаем: .

Изобразим графически на рисунке 2 оценку корреляционной функции:

*Рисунок 2. Графическая оценка НКФ*

# Построение модели АР

Построим модели авторегрессии АР(М) = АРСС(М, 0) порядков M = 1, 2, 3. Модель авторегрессии имеет вид:

гдевыходная случайная последовательность с корреляционной функцией , – входная некоррелированная случайная последовательность с нулевым математическим ожиданием и единичной дисперсией.

Пусть M=1



Получаем ,

Проверяем модель на устойчивость:

⇒модель устойчива.

Пусть M=2



Получаем , ,

Проверяем модель на устойчивость:

⇒модель устойчива.

Пусть M=3



Получаем

Проверяем модель на устойчивость:

0.983229231996

⇒модель устойчива.

Расчёт теоретической НКФ выходной последовательности

Для каждой модели рассчитываем теоретические нормированные корреляционные функции выходной последовательности. Недостающие значения нормированной корреляционной функции отыщем из системы:

,

в которой первые (N+M+1) значений нам уже известно и совпадают со значениями выборочной корреляционной функции входной последовательности.

Выбор наилучшей модели СП в классе моделей АР

Выбор наилучшей модели проводиться на основании анализа нормированных корреляционных функций с использованием критерия среднего квадратичного отклонения по первым десяти отсчетам:

,

где  - рассчитанная теоретическая нормированная корреляционная функция исходного процесса;

-выборочная нормированная корреляционная функция исходного процесса.

В таблицу 2 занесем полученные средние квадратичные отклонения и проанализируем их. Жирным шрифтом выделим лучшую модель АР(М).

*Таблица 2 – Модели* ***авторегрессии*** *АР(М)*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Порядок  модели | Параметры модели  , | | | | Погрешность  модели | |
| M |  |  |  |  | |  |
| 1 |  |  |  |  | | 0.07566 |
| 2 |  |  |  |  | | 0.02955 |
| 3 |  |  |  |  | | **0.00324** |

# Построение моделей скользящего среднего

Построим модели скользящего среднего СС(*N*) = АРСС(0, *N*) порядков *N* = 0, 1, 2, 3 на основе решения системы нелинейных уравнений.

Учитывая, что M=0, получаем:



Для каждого из рассматриваемых N, составляем систему уравнений. А затем, решим составленные системы, найдя тем самым, оценки искомых коэффициентов, подобно тому, как мы это делали, в случаи нахождения оценки коэффициентов АР. Результаты построения модели заносим в таблицу. Модель APCC (0,N) = СС (N) устойчива всегда.

Пусть N=0:



Пусть N=1:



Пусть N=2:



Пусть N=3:



Заносим средние квадратичные отклонения в таблицу 3 и анализируем их. В таблице жирным отмечена лучшая модель.

*Таблица 3 – Результат построения моделей СС*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Порядок  модели СС(N) | Параметры модели | | | | |
| N |  |  |  |  |  |
| 0 | 5.9541 |  |  |  | 0.1233 |
| 1 | 5.7253 | - 0.0177 |  |  | 0.0536 |
| 2 | 5.6645 | - 0.0174 | 1.0099 |  | 0.0491 |
| 3 | 5.6002 | - 0.0175 | 0.0148 | 0.0073 | **0.0177** |

Расчёт теоретической КФ на примере лучшей модели CC

Для модели СС(3):



Подставим числовые значения:

Выражаем все коэффициенты, и решаем систему с помощью метода простых итераций:

**4. Смешанные модели.**

Построим модели авторегрессии скользящего среднего АРСС(*M,N*)

, порядков *M* и *N* не превышающих 3. Здесь – входная некоррелированная случайная последовательность с нулевым математическим ожиданием и единичной дисперсией, а – выходная случайная последовательность с корреляционной функцией .

Для построения модели АРСС существует два метода. Рассмотрим каждый из них.

Метод 1:

1. Из уравнений

находим оценки неизвестных параметров .

1. Из уравнений

находим значения оставшихся параметров .

Решим для модели АРСС(3,2):

Найдем коэффициенты системы линейных уравнений.

Модель является устойчивой т.к.

Подставляем в систему:

Откуда:

Метод 2:

1) Из уравнений Бнаходим оценки неизвестных параметров .

2) Построим промежуточную последовательность по следующей формуле: и запишем для неё систему уравнений.

3) Найдем выборочную корреляционную функцию промежуточной случайной последовательности .

4) Из системы (N+1) нелинейных уравнений находятся оценки (N+1) неизвестных коэффициентов .

Для модели АРСС(3,2):

Находим оценки неизвестных параметров (аналогично методу 1):

Модель является устойчивой т.к.

Строим промежуточную случайную последовательность с помощью уравнения и находим выборочную корреляционную функцию по этой последовательности :

Далее найдем из системы нелинейных уравнений:

Откуда:

Таблица 4 – Смешанные модели АРСС(M,N) (Метод 1)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Порядок модели | | Параметры модели | | | | | | | | Погреш-ность модели |
| M | N |  | |  |  |  |  |  |  |  |
| 1 | 1 | 0.2555 | |  |  | 5.6501 | -3.2592 |  |  | 0.054661 |
| 1 | 2 | Модель неустойчива | | | | | | | | - |
| 1 | 3 | -0.5574 |  | |  | 5.6118 | 1.5883 | - 1.1136 | 0.8819 | 0.018982 |
| 2 | 1 | Модель не существует | | | | | | | | 0.071930 |
| 2 | 2 | -0.7409 | -0.4820 | |  | 5.6288 | 2.5895 | 1.379 |  | 0.000515 |
| 2 | 3 | -0.7425 | -0.4862 | |  | 5.6287 | 2.5985 | 1.4008 | -0.0077 | 0.000478 |
| 3 | 1 | -0.4670 | | -0.1637 | 0.1025 | 5.6307 | 1.0499 |  |  | 0.003251 |
| 3 | 2 | -0.7481 | | -0.4903 | -0.0027 | 5.6287 | 2.6299 | 1.4152 |  | **0.000478** |
| 3 | 3 | Модель неустойчива | | | | | | | | **-** |

Таблица 5 – Смешанные модели АРСС(M,N) (Метод 2)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Порядок модели | | Параметры модели | | | | | | | | Погреш-ность модели |
| M | N |  | |  |  |  |  |  |  |  |
| 1 | 1 | 0.2555 | |  |  | 5.6501 | -3.2586 |  |  | 0.054661 |
| 1 | 2 | Модель неустойчива | | | | | | | | - |
| 1 | 3 | -0.5574 |  | |  | 4.4227 | 3.3808 | -1.9533 | 1.1201 | 0.018982 |
| 2 | 1 | Модель не существует | | | | | | | | 0.071930 |
| 2 | 2 | -0.7409 | -0.4820 | |  | 1.3775 | 2.5912 | 5.6380 |  | 0.000515 |
| 2 | 3 | -0.7425 | -0.4862 | |  | -0.0056 | 1.3982 | 2.6001 | 5.6383 | 0.000478 |
| 3 | 1 | -0.4670 | | -0.1637 | 0.1025 | 1.0492 | 5.6332 |  |  | 0.003251 |
| 3 | 2 | -0.7481 | | -0.4903 | -0.0027 | 4.4846 | 2.007 | 1.7763 |  | **0.000478** |
| 3 | 3 | Модель неустойчива | | | | | | | | **-** |

Приложение А: Текст программы

Задание 1

private final String path = "data.txt";  
private final int COUNT\_N=5000;  
double rk[] = new double[11];  
double Rn[]=new double[11];  
private List<Double> data = new ArrayList<>();  
public TspMain(){  
 fillList();  
 System.*out*.println("Mx = "+Mx(COUNT\_N));  
 System.*out*.println("S^2="+S2(COUNT\_N,Mx(COUNT\_N)));  
 System.*out*.println("S^2 fix="+S2fix(COUNT\_N,Mx(COUNT\_N)));  
 for(int i = 0;i<11;i++){  
 Rn[i]=Rk(COUNT\_N,i,Mx(COUNT\_N));  
 rk[i] = rk(COUNT\_N,i,Mx(COUNT\_N));  
 System.*out*.println(rk(COUNT\_N,i,Mx(COUNT\_N)));  
 }  
 //Tkop(COUNT\_N,Mx(COUNT\_N));  
 AR ar = new AR(rk);  
 ar.calcAR1(new double[][]{  
 {Rk(COUNT\_N,1,Mx(COUNT\_N)),1,Rk(COUNT\_N,0,Mx(COUNT\_N))},  
 {Rk(COUNT\_N,0,Mx(COUNT\_N)),0,Rk(COUNT\_N,1,Mx(COUNT\_N))}  
 });  
 ar.calcAR2(new double[][]{  
 {Rk(COUNT\_N,1,Mx(COUNT\_N)),Rk(COUNT\_N,2,Mx(COUNT\_N)),1,Rk(COUNT\_N,0,Mx(COUNT\_N))},  
 {Rk(COUNT\_N,0,Mx(COUNT\_N)),Rk(COUNT\_N,1,Mx(COUNT\_N)),0,Rk(COUNT\_N,1,Mx(COUNT\_N))},  
 {Rk(COUNT\_N,1,Mx(COUNT\_N)),Rk(COUNT\_N,0,Mx(COUNT\_N)),0,Rk(COUNT\_N,2,Mx(COUNT\_N))}  
  
 });  
 ar.calcAR3(new double[][]{  
 {Rk(COUNT\_N,1,Mx(COUNT\_N)),Rk(COUNT\_N,2,Mx(COUNT\_N)),Rk(COUNT\_N,3,Mx(COUNT\_N)),1,Rk(COUNT\_N,0,Mx(COUNT\_N))},  
 {Rk(COUNT\_N,0,Mx(COUNT\_N)),Rk(COUNT\_N,1,Mx(COUNT\_N)),Rk(COUNT\_N,2,Mx(COUNT\_N)),0,Rk(COUNT\_N,1,Mx(COUNT\_N))},  
 {Rk(COUNT\_N,1,Mx(COUNT\_N)),Rk(COUNT\_N,0,Mx(COUNT\_N)),Rk(COUNT\_N,1,Mx(COUNT\_N)),0,Rk(COUNT\_N,2,Mx(COUNT\_N))},  
 {Rk(COUNT\_N,2,Mx(COUNT\_N)),Rk(COUNT\_N,1,Mx(COUNT\_N)),Rk(COUNT\_N,0,Mx(COUNT\_N)),0,Rk(COUNT\_N,3,Mx(COUNT\_N))}  
  
 });  
 SS ss = new SS(Rn,rk);  
  
 ss.calcSS0();  
 ss.calcSS1();  
 ss.calcSS2();  
 ss.calcSS3();  
 ARSS arss = new ARSS(Rn);  
}  
  
public void calcEps2(){  
 double eps=0;  
 double[] r = new double[11];  
 r[0]=1;  
 r[1]=-0.2639;  
 r[2]=-0.4501;  
 for(int i = 0;i<9;i++){  
 if(i!=0&&i!=1&&i!=2)  
 r[i]=r[i-1]\*(-0.3028)+r[i-2]\*(- 0.1473);  
 //System.out.println(i+": "+r[i]);  
 eps+=Math.*pow*(r[i]-rk(COUNT\_N,i,Mx(COUNT\_N)),2);  
 }  
 System.*out*.println(eps);  
}  
public void calcEps3(){  
 double eps=0;  
 double[] r = new double[11];  
 r[0]=1;  
 r[1]=-0.2639;  
 r[2]=-0.4501;  
 r[3]=-0.2838-0.1021-0.1295;  
 for(int i = 0;i<9;i++){  
 if(i!=0&&i!=1&&i!=2&&i!=3)  
 r[i]=r[i-1]\*(-0.2838)+r[i-2]\*(-0.1021)+r[i-3]\*(-0.1295);  
 //System.out.println(i+": "+r[i]);  
 eps+=Math.*pow*(r[i]-rk(COUNT\_N,i,Mx(COUNT\_N)),2);  
 }  
 System.*out*.println(eps);  
  
}  
  
public void fillList() {  
 try {  
  
  
 Scanner sc = new Scanner(new File(path));  
 for (int i = 0; i < COUNT\_N; i++) {  
 //System.out.println(sc.nextLine());  
 data.add(Double.*valueOf*(sc.nextLine()));  
 }  
  
 } catch (FileNotFoundException e) {  
 System.*out*.println("Файла не существует");  
 }  
}  
public double Mx(int n){  
 double res=0;  
 for(Double x:data){  
 res+=x;  
 }  
 return res/n;  
  
  
  
}  
public double S2(int n,double mx){  
 double res=0;  
 for(Double x:data){  
 res+=Math.*pow*((x-mx),2);  
  
 }  
 return res/n;  
}  
public double S2fix(int n,double mx){  
 double res=0;  
 for(Double x:data){  
 res+=Math.*pow*((x-mx),2);  
  
 }  
 return res/(n-1);  
}  
public double Rk(int n,int k,double mx){  
 double res=0;  
 for(int i = 0;i<n-k;i++){  
 res+=(data.get(i)-mx)\*(data.get(i+k)-mx);  
 }  
 return res/(n-k-1);  
  
  
}  
public double rk(int n,int k,double mx){  
 return Rk(n,k,mx)/Rk(n,0,mx);  
}  
public void Tkop(int n,double mx){  
 double T;  
 for(int m = 0;m<100;m++)  
 if(Math.*abs*(rk(n,m,mx))<Math.*pow*(Math.*E*,-1)){  
 System.*out*.println(m);  
  
 }

Задание 3

private final int COUNT\_N=5000;  
private double[] NCF1 = new double[11];  
private double[] NCF2 = new double[11];  
private double[] NCF3 = new double[11];  
private double rk[];  
public AR(double[] rk){  
 this.rk = rk;  
  
}  
public void calcAR1(double[][] a){  
 double[] betas = new double[1];  
 double x[]=new double[a.length];  
 for (int i = 0; i < x.length; i++) {  
 x[i] = a[i][a[i].length - 1];  
 }  
 double m;  
 for (int k = 1; k < a.length; k++) {  
 for (int j = k; j < a.length; j++) {  
 m = a[j][k - 1] / a[k - 1][k - 1];  
 for (int i = 0; i < a[j].length; i++) {  
 a[j][i] = a[j][i] - m \* a[k - 1][i];  
 }  
 x[j] = x[j] - m \* x[k - 1];  
 }  
 }  
  
  
  
 for (int i=a.length-1;i>=0;i--) {  
 for (int j=i+1;j<a.length;j++) x[i]-=a[i][j]\*x[j];  
 x[i] = x[i] / a[i][i];  
 }  
  
 System.*out*.println(("b = "+x[0]));  
  
 System.*out*.println("a = "+Math.*sqrt*(x[x.length-1]));  
 calcEps1(new double[]{x[0]});  
  
}  
public void calcEps1(double[] betas){  
 NCF1[0]= 1;  
 NCF1[1]= rk[1];  
 for(int i = 2;i<NCF1.length;i++){  
 NCF1[i] = betas[0] \* NCF1[i-1];  
 }  
 double eps=0;  
 for(int i =1;i<NCF1.length;i++){  
 eps += Math.*pow*(NCF1[i] - rk[i], 2);  
 }  
 System.*out*.println("eps = "+eps);  
  
}  
public void calcAR2(double[][] a){  
 System.*out*.println("AR2");  
 double x[]=new double[a.length];  
 for (int i = 0; i < x.length; i++) {  
 x[i] = a[i][a[i].length - 1];  
 }  
 double m;  
 for (int k = 1; k < a.length; k++) {  
 for (int j = k; j < a.length; j++) {  
 m = a[j][k - 1] / a[k - 1][k - 1];  
 for (int i = 0; i < a[j].length; i++) {  
 a[j][i] = a[j][i] - m \* a[k - 1][i];  
 }  
 x[j] = x[j] - m \* x[k - 1];  
 }  
 }  
  
  
 for (int i=a.length-1;i>=0;i--) {  
 for (int j=i+1;j<a.length;j++) x[i]-=a[i][j]\*x[j];  
 x[i] = x[i] / a[i][i];  
 }  
  
 for (int i = 0; i < x.length-1; i++) {  
 System.*out*.println("b("+i+") = "+x[i]);  
 }  
  
 System.*out*.println("a = "+Math.*sqrt*(x[x.length-1]));  
 calcEps2(new double[]{x[0],x[1]});  
  
}  
public void calcEps2(double[] betas){  
 NCF2[0]= 1;  
 NCF2[1]= rk[1];  
 NCF2[2]= rk[2];  
 for(int i = 3;i<NCF2.length;i++){  
 NCF2[i] = betas[0] \* NCF2[i-1]+betas[1] \* NCF2[i-2];  
 }  
 double eps=0;  
 for(int i =1;i<NCF2.length;i++){  
 eps += Math.*pow*(NCF2[i] - rk[i], 2);  
 }  
 System.*out*.println("eps = "+eps);  
  
}  
public void calcAR3(double [][]a){  
 System.*out*.println("AR3");  
 double x[]=new double[a.length];  
 for (int i = 0; i < x.length; i++) {  
 x[i] = a[i][a[i].length - 1];  
 }  
 double m;  
 for (int k = 1; k < a.length; k++) {  
 for (int j = k; j < a.length; j++) {  
 m = a[j][k - 1] / a[k - 1][k - 1];  
 for (int i = 0; i < a[j].length; i++) {  
 a[j][i] = a[j][i] - m \* a[k - 1][i];  
 }  
 x[j] = x[j] - m \* x[k - 1];  
 }  
 }  
  
  
  
 for (int i=a.length-1;i>=0;i--) {  
 for (int j=i+1;j<a.length;j++) x[i]-=a[i][j]\*x[j];  
 x[i] = x[i] / a[i][i];  
 }  
  
 for (int i = 0; i < x.length-1; i++) {  
 System.*out*.println("b("+i+") = "+x[i]);  
 }  
 System.*out*.println("a = "+Math.*sqrt*(x[x.length-1]));  
 calcEps3(new double[]{x[0],x[1],x[2]});  
  
}  
public void calcEps3(double[] betas){  
 NCF3[0]= 1;  
 NCF3[1]= rk[1];  
 NCF3[2]= rk[2];  
 NCF3[3]= rk[3];  
 for(int i = 4;i<NCF1.length;i++){  
 NCF3[i] = betas[0] \* NCF3[i-1]+betas[1] \* NCF3[i-2]+betas[2] \* NCF3[i-3];  
 }  
 double eps=0;  
 for(int i =1;i<NCF3.length;i++){  
 eps += Math.*pow*(NCF3[i] - rk[i], 2);  
 }  
 System.*out*.println("eps = "+eps);  
  
}

Задание 3

private double Rn[];  
private double rk[];  
private double[] NCF0 = new double[11];  
private double[] NCF1 = new double[11];  
private double[] NCF2 = new double[11];  
private double[] NCF3 = new double[11];  
public SS(double[] Rn,double rk[]){  
 this.Rn=Rn;  
 this.rk=rk;  
  
  
}  
public void calcSS0(){  
 System.*out*.println("SS 0");  
 System.*out*.println("a = "+Math.*sqrt*(Rn[0]) );  
 NCF0[0]= 1;  
 NCF0[1]= 0;  
 for(int i = 2;i<NCF0.length;i++){  
 NCF0[i] = 0;  
 }  
 double eps=0;  
 for(int i =1;i<NCF0.length;i++){  
 eps += Math.*pow*(NCF0[i] - rk[i], 2);  
 }  
 System.*out*.println("eps = "+eps);  
  
}  
public void calcSS1() {  
 System.*out*.println("SS 1");  
 double alpha0 = Math.*sqrt*(Rn[0]);  
 double alpha1 = 0;  
  
 double eps = 0.0001;  
  
 while (Math.*abs*(Math.*pow*(alpha0, 2) + Math.*pow*(alpha1, 2) - Rn[0]) > eps ||  
 Math.*abs*(alpha0\*alpha1-Rn[1]) > eps) {  
 alpha1 = Rn[1]/alpha0;  
 if(Rn[0] < Math.*pow*(alpha1, 2)) {  
 return;  
 } else {  
 alpha0 = Math.*sqrt*(Rn[0]-Math.*pow*(alpha1, 2));  
 }  
 }  
  
 System.*out*.println(  
 "a0 = "+alpha0  
 );  
 System.*out*.println(  
 "a1 = "+alpha1  
 );  
 NCF1[0]= 1;  
 NCF1[1]= rk[1];  
 for(int i = 2;i<NCF1.length;i++){  
 NCF1[i] = 0;  
 }  
 double epsCalc=0;  
 for(int i =1;i<NCF1.length;i++){  
 epsCalc += Math.*pow*(NCF1[i] - rk[i], 2);  
 }  
 System.*out*.println("eps = "+epsCalc);  
}  
  
public void calcSS2() {  
 System.*out*.println("SS 2");  
 double alpha0 = Math.*sqrt*(Rn[0]);  
 double alpha1 = 0;  
 double alpha2 = 0;  
  
 double eps = 0.0001;  
  
 while (Math.*abs*(Math.*pow*(alpha0, 2) + Math.*pow*(alpha1, 2) + Math.*pow*(alpha2, 2) - Rn[0]) > eps ||  
 Math.*abs*(alpha0\*alpha1+alpha1\*alpha2-Rn[1]) > eps ||  
 Math.*abs*(alpha0\*alpha2-Rn[2]) > eps) {  
 alpha2 = Rn[2]/alpha0;  
 alpha1 = (Rn[1]-alpha1\*alpha2)/alpha0;  
 if(Rn[0] < Math.*pow*(alpha1, 2)+Math.*pow*(alpha2, 2)) {  
 return;  
 } else {  
 alpha0 = Math.*sqrt*(Rn[0]-Math.*pow*(alpha1, 2)-Math.*pow*(alpha2, 2));  
 }  
 }  
 System.*out*.println(  
 "a0 = "+alpha0  
 );  
 System.*out*.println(  
 "a1 = "+alpha1  
 );  
 System.*out*.println(  
 "a2 = "+alpha2  
 );  
 NCF2[0]= 1;  
 NCF2[1]= rk[1];  
 NCF2[2]= rk[2];  
 for(int i = 3;i<NCF2.length;i++){  
 NCF2[i] = 0;  
 }  
 double epsCalc=0;  
 for(int i =1;i<NCF2.length;i++){  
 epsCalc += Math.*pow*(NCF2[i] - rk[i], 2);  
 }  
 System.*out*.println("eps = "+epsCalc);  
  
}  
  
public void calcSS3() {  
 System.*out*.println("SS 3");  
 double alpha0 = Math.*sqrt*(Rn[0]);  
 double alpha1 = 0;  
 double alpha2 = 0;  
 double alpha3 = 0;  
  
 double eps = 0.0001;  
  
 while (Math.*abs*(Math.*pow*(alpha0, 2) + Math.*pow*(alpha1, 2) + Math.*pow*(alpha2, 2) + Math.*pow*(alpha3, 2) - Rn[0]) > eps ||  
 Math.*abs*(alpha0\*alpha1+alpha1\*alpha2+alpha2\*alpha3-Rn[1]) > eps ||  
 Math.*abs*(alpha0\*alpha2+alpha1\*alpha3-Rn[2]) > eps ||  
 Math.*abs*(alpha0\*alpha3-Rn[3]) > eps) {  
 alpha3 = Rn[3]/alpha0;  
 alpha2 = (Rn[2]-alpha1\*alpha3)/alpha0;  
 alpha1 = (Rn[1]-alpha1\*alpha2-alpha2\*alpha3)/alpha0;  
 if(Rn[0] < Math.*pow*(alpha1, 2)+Math.*pow*(alpha2, 2)+Math.*pow*(alpha3, 2)) {  
 return;  
 } else {  
 alpha0 = Math.*sqrt*(Rn[0]-Math.*pow*(alpha1, 2)-Math.*pow*(alpha2, 2)-Math.*pow*(alpha3, 2));  
 }  
 }  
 System.*out*.println(  
 "a0 = "+alpha0  
 );  
 System.*out*.println(  
 "a1 = "+alpha1  
 );  
 System.*out*.println(  
 "a2 = "+alpha2  
 );  
 System.*out*.println(  
 "a3 = "+alpha3  
 );  
 NCF3[0]= 1;  
 NCF3[1]= rk[1];  
 NCF3[2]= rk[2];  
 NCF3[3]= rk[3];  
 for(int i = 4;i<NCF3.length;i++){  
 NCF3[i] = 0;  
 }  
 double epsCalc=0;  
 for(int i =1;i<NCF3.length;i++){  
 epsCalc += Math.*pow*(NCF3[i] - rk[i], 2);  
 }  
 System.*out*.println("eps = "+epsCalc);  
}

Задание 4

private double[] R;  
  
public ARSS(double[] r) {  
 R = r;  
   
  
  
 System.*out*.println("M3N2");

double[] alp32 = M3N2();  
 System.*out*.println("alpha0=" + alp32[0]);  
 System.*out*.println("alpha1=" + alp32[1]);  
 System.*out*.println("alpha2=" + alp32[2]);  
 M3N2M2();

}

public double[] M3N2() {  
  
 double[][] a = new double[][]{  
 {R[2], R[1], R[0], R[3]},  
 {R[3], R[2], R[1], R[4]},  
 {R[4], R[3], R[2], R[5]}  
 };  
 double[] x = solveSystem(a);  
 double[] alpha = new double[3];  
 alpha[0] = Math.*sqrt*(R[0]);  
 int i = 0;  
 double b1 = x[0];  
 double b2 = x[1];  
 double b3 = x[2];  
 while (i < 100) {  
 alpha[2] = (R[2] - b1 \* R[1] - b2 \* R[0] - b3 \* R[1]) / alpha[0];  
 alpha[1] = (R[1] - b1 \* R[0] - b2 \* R[1] - b3 \* R[2] - alpha[2] \* b1 \* alpha[0]) / ((alpha[0] + alpha[2]));  
  
 double tmp = R[0] - b1 \* R[1] - b2 \* R[2] - b3 \* R[3] - alpha[1] \* (b1 \* alpha[0] + alpha[1]) - alpha[2] \* (b1 \* (b1 \* alpha[0] + alpha[1]) + b2 \* alpha[0] + alpha[2]);  
 if (tmp < 0) return new double[]{0};  
 else alpha[0] = Math.*sqrt*(tmp);  
 i++;  
 }  
 return alpha;  
  
}  
public double[] M3N2M2(){  
 double[][] a = new double[][]{  
 {R[2], R[1], R[0], R[3]},  
 {R[3], R[2], R[1], R[4]},  
 {R[4], R[3], R[2], R[5]}  
 };  
 double[] x = solveSystem(a);  
 int i = 0;  
  
 double alpha1 = 0;  
 double alpha2 = 0;  
 double alpha3 = 0;  
  
 double eps = 0.0001;  
 double b1 = x[0];  
 double b2 = x[1];  
 double b3 = x[2];  
 double [] res=new double[10];  
 for(int m = 0;m<10;m++){  
 res[m]=R[m]-b1\*R[Math.*abs*(m-1)]-b2\*R[Math.*abs*(m-2)]-b3\*R[Math.*abs*(m-3)];  
 System.*out*.println("R["+m+"]="+R[m]+"-"+b1+" "+ R[Math.*abs*(m-1)]+"-"+b2+" "+R[Math.*abs*(m-2)]+"-"+b3+" "+R[Math.*abs*(m-3)]);  
 }  
 double alpha0 = Math.*sqrt*(res[0]);  
 while (Math.*abs*(Math.*pow*(alpha0, 2) + Math.*pow*(alpha1, 2) + Math.*pow*(alpha2, 2) + Math.*pow*(alpha3, 2) - res[0]) > eps ||  
 Math.*abs*(alpha0\*alpha1+alpha1\*alpha2+alpha2\*alpha3-res[1]) > eps ||  
 Math.*abs*(alpha0\*alpha2+alpha1\*alpha3-res[2]) > eps ||  
 Math.*abs*(alpha0\*alpha3-res[3]) > eps) {  
 alpha3 = res[3]/alpha0;  
 alpha2 = (res[2]-alpha1\*alpha3)/alpha0;  
 alpha1 = (res[1]-alpha1\*alpha2-alpha2\*alpha3)/alpha0;  
 if(res[0] < Math.*pow*(alpha1, 2)+Math.*pow*(alpha2, 2)+Math.*pow*(alpha3, 2)) {  
 return new double[4];  
 } else {  
 alpha0 = Math.*sqrt*(res[0]-Math.*pow*(alpha1, 2)-Math.*pow*(alpha2, 2)-Math.*pow*(alpha3, 2));  
 }  
 }  
 System.*out*.println("a0="+alpha0);  
 System.*out*.println("a1="+alpha1);  
 System.*out*.println("a2="+alpha2);  
 System.*out*.println("a3="+alpha3);  
  
 return res;  
}

}