Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования "Уфимский государственный авиационный технический университет"

Кафедра Высокопроизводительных вычислительных технологий и систем

Дисциплина: Математическая статистика

Отчет по лабораторной работе № 3

Тема: «Использование цепей Маркова при моделировании»

Группа ПМ-453	Фамилия И.О.	Подпись	Дата	Оценка
Студент	Шамаев И.Р.			
Принял	Маякова С.А.			

Уфа 2022

Теоретический материал:

Марковский процесс является вероятностным аналогом процесса в классической механике. Марковский процесс (цепь) характеризуется отсутствием памяти, т.е. статистические характеристики ближайшего будущего определяются исключительно настоящим, без учета прошлого.

Пусть происходит случайный процесс в системе X с дискретными состояниями, которые она может принимать в последовательности шагов с номерами $0, 1, 2, ..., k, ...$ Обозначим состояние системы как $x=(x_1,\dots,x_N)$, где N - число степеней свободы. Множество состояний системы образует доступное системе фазовое пространство Ω. Пусть случайный процесс представляет собой последовательность событий вида $\{X(k)=x_i\}$ $(i=1, 2, \dots, n;\ k=0, 1, 2, \dots)$. Наиболее важной характеристикой этой последовательности событий являются вероятности состояния системы $P(X(k)=x_i)$ $(i=1, 2, \dots, n;\ k=0, 1, 2, \dots)$, где $P(X(k)=x_i)$ — вероятность того, что на k-ом шаге система X будет находиться в состоянии x_i. Формальное определение марковской цепи выглядит следующим образом.

Определение 1

Процесс, определяемый последовательностью $\{X(k)=x_i\}$ $(i=1, 2, \dots, n;\ k=0, 1, 2, \dots)$, называется *марковским процессом с дискретными состояниями и дискретным временем* (или *марковской цепью*), если для любого фиксированного момента времени условные вероятности состояний системы в будущем зависят только от состояний системы в настоящем и не зависят от того, когда и откуда система пришла в это состояние:

$$P(X(k+1)=x_{i_{k+1}} \mid X(k)=x_{i_k},\ X(k-1)=x_{i_{k-1}},\ \dots,\ X(0)=x_{i_0}) =$$

$$= P(X(k+1)=x_{i_{k+1}} \mid X(k)=x_{i_k}).$$

Обозначим $P_{ij}(k)$ вероятность перехода системы X из состояния x_i в состояние x_j на k-ом шаге: $P_{ij}(k)=P(X(k)=x_j \mid X(k-1)=x_i)$.

Цепь Маркова называется *однородной*, если переходные вероятности $P_{ij}(k)$ не зависят от номера шага k: $P_{ij}(k)=P_{ij}$. Далее будем рассматривать только однородные цепи Маркова.

При некоторых условиях в цепи Маркова с возрастанием k устанавливается стационарный режим, в котором система X продолжает переходить из одного состояния в другое, но вероятности этих состояний уже от номера шага не зависят. Такое распределение вероятности называется *предельным* или *стационарным* распределением. Для существования стационарного распределения вероятности $u_i=\lim\limits_{k\to\infty} P(X(k)=x_i)$ необходимо выполнение следующих условий:

1. для всех i $u_i>0$,

2. $\sum_i u_i = 1$,
3. $u_j = \sum_i u_i P_{ij}$.

Если сгенерированные состояния являются равновесными состояниями рассматриваемой системы, должно выполняться условие эргодичности. Чтобы дать его точное определение в контексте марковских цепей, нужно сделать ряд замечаний. Пусть $P_{ij}^{(n)}$ обозначает вероятность перехода из состояния $\underline{x}_i$ в состояние $\underline{x}_j$ за n шагов, т. е. вероятность *n*-шагового перехода.

Состояние $\underline{x}_j$ может быть получено из $\underline{x}_i$, если существует такое $n>0$, что $P_{ij}^{(n)}>0$. Цепь является *неприводимой* только в том случае, когда каждое ее состояние может быть получено из каждого другого ее состояния. Это, очевидно, будет необходимо для применения метода. Если же цепочка является *приводимой*, последовательность состояний будет разбиваться на классы, между которыми переходы отсутствуют.

Определение 2

Состояние $\underline{x}_i$ имеет *период* $t>1$, если $P_{ii}^{(n)}=0$ до тех пор, пока $n=zt$ кратно t, а t - наибольшее целое число с этим свойством. Состояние является *апериодическим*, если такого t не существует.

Пусть $f_{ij}^{(n)}$ обозначает вероятность того, что в процессе, стартующем из $\underline{x}_i$, первый переход в $\underline{x}_j$ осуществляется на *n*-м шаге. Кроме того, пусть

$$f_{ii}^{(0)}=0,\ f_{ij}=\sum_{n=1}^{\infty} f_{ij}^{(n)},\ \mu_i=\sum_{n=1}^{\infty} n f_{ij}^{(n)}.$$

Тогда f_{ij} есть вероятность того, что, стартуя из состояния $\underline{x}_i$ системы пройдет через состояние $\underline{x}_j$. В случае $f_{ij}=1$ состояние $\underline{x}_i$ называется *устойчивым*, а μ_i - *средним возвратным временем*.

Теперь приведем точное понятие эргодичности.

Определение 3

Состояние $\underline{x}_i$ называется *эргодическим*, если оно является апериодическим и устойчивым с конечным средним временем возврата. Марковская цепь, состоящая только из эргодических элементов, называется эргодической.

Для приложения теории марковских цепей к математическому моделированию следующая теорема является основной.

Теорема 1

Неприводимая апериодическая цепь имеет инвариантное распределение тогда и только тогда, когда она является эргодической. В этом случае $u_i>0$ для любых i и безусловные вероятности стремятся к u_i вне зависимости от начального распределения.

Эта теорема гарантирует, что состояния марковской цепи, в конечном счете, распределятся в соответствии с некоторым предельным распределением. Более того, начальное распределение при этом не играет роли.

Практическая часть:

Задание 1.

Согласно индивидуальному варианту задать начальные вероятности $P(0)=(a1,a2,a3)^T$ и одношаговую матрицу переходных вероятностей $\pi=\|p_{ij}\|$. Определить, количество залпов n, необходимое для достижения предельных вероятностей, рассчитанных согласно системе (*2), т.е. такое n, при котором $P(n) \approx (p_1, p_2, p_3)^T$ с точностью до 10^{-4}, где $P(n)=(\pi^T)^n \times P(0)$ – вероятность событий (A_1, A_2, A_3) на n-ом временном шаге Какими свойствами должна обладать матрица π, чтобы предельные вероятности существовали? Насколько вероятным является поражение цели на n-ом шаге? Как должна выглядеть матрица π, чтобы предельные вероятности были равны $p_1=0$, $p_2=0$, $p_3=1$?

Индивидуальное задание

1. Начальные вероятности:
   $a1=1/(n+1)$, где n – номер по списку
   $a2$, $a3$ генерируются случайно, соблюдая условие $a2+a3=1-a1$
2. Матрица переходных вероятностей:

$$\pi=\begin{pmatrix} 0.5 & 0.25 & 0.25 \\ 0.5-1/(10+n) & 0 & 0.5+1/(10+n) \\ 0.25 & 0.25 & 0.5 \end{pmatrix}$$

Листинг программы в Приложении.

Пример выполнения программы:

```
Original matrix Pi
0.5     0.25    0.25
0.456522        0       0.543478
0.25    0.25    0.5

Transposed matrix
0.5     0.456522        0.25
0.25    0       0.25
0.25    0.543478        0.5
( 0.388406 ; 0.2 ; 0.411594 )

( 0.388328 ; 0.200051 ; 0.41162 )

n = 5
```

Было определено количество залпов: $n=5$.

Какими свойствами должна обладать матрица π, чтобы предельные вероятности существовали?

Для существования стационарного распределения вероятности $u_i=\lim\limits_{k\to\infty} P(X(k)=x_i)$ необходимо выполнение следующих условий:

1) для всех i $u_i>0$,

2) $\sum_i u_i = 1$,

3) $u_j = \sum_i u_i P_{ij}$.

Насколько вероятным является поражение цели на n-ом шаге?

Различным методам корректировки соответствуют различные вероятности указанных событий на залп, задаваемые матрицами перехода вида

$$\|p_{ij}\| = \begin{Vmatrix} w_1 & w_2 & w_3 \\ v_1 & v_2 & v_3 \\ z_1 & z_2 & z_3 \end{Vmatrix}.$$

w_1 – вероятность наступления в результате залпа события A_1, если в результате предыдущего залпа наступило событие A_1;

w_2 – вероятность наступления в результате залпа события A_2, если в результате предыдущего залпа наступило событие A_1;

w_3 – вероятность наступления в результате залпа события A_3, если в результате предыдущего залпа наступило событие A_1;

v_1 – вероятность наступления в результате залпа события A_1, если в результате предыдущего залпа наступило событие A_2;

v_2 – вероятность наступления в результате залпа события A_2, если в результате предыдущего залпа наступило событие A_2;

v_3 – вероятность наступления в результате залпа события A_3, если в результате предыдущего залпа наступило событие A_2;

z_1 – вероятность наступления в результате залпа события A_1, если в результате предыдущего залпа наступило событие A_3;

z_2 – вероятность наступления в результате залпа события A_2, если в результате предыдущего залпа наступило событие A_3;

z_3 – вероятность наступления в результате залпа события A_3, если в результате предыдущего залпа наступило событие A_3;

Как должна выглядеть матрица π, чтобы предельные вероятности были равны $p_1=0$, $p_2=0$, $p_3=1$?

$$\pi = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Задание 2.

Выбрать регион для прогнозирования численности населения (см. Табл. 1). Реализовать модель прогнозирования численности населения согласно уравнению (3). Коэффициенты рождаемости, смертности, матрицу переходной вероятности, стартовое значение численности населения взять из соответствующего выбранному региону раздела (для инициализации начальных значений частных мат. ожиданий $M_s(0)$ с s=1,2,3 стартовое значение численности населения нужно разделить на три части, например, в долях, пропорциональных коэффициенту рождаемости[1]). Построить график мат. ожидания численности населения M(h) в зависимости от номера шага h. Сравнить его с графиком, приведенным в разделе, соответствующем выбранному региону.

Регион: *Европа и Средняя Азия.*

.

$$X_{n+1}=\alpha X_n, \alpha=\frac{1000+(K_p-K_c)}{1000} \qquad (2)$$

Где K_c – коэффициент смертности, K_p- коэффициент рождаемости.

Разностное уравнение для частных первых моментов решения уравнения (2) имеет вид (3):

$$M_j(h+1)=\sum_{x=1}^{3}\pi_{js}a_s M_s(h), M(h)=\sum_{s=1}^{3}M_s(h), (j=1,2,3; h=0,1,2,\ldots), \qquad (3)$$

где h - шаг изменения времени, π_{js} – элементы матрицы переходных вероятностей, a_s– коэффициенты уравнения Мальтуса (2).

Европа и Средняя Азия.

	Коэффициент рождаемости, ‰	*Коэффициент смертности, ‰*
максимальный	*22*	*15*
средний	*12*	*11*
минимальный	*9*	*5*

Матрица переходных вероятностей

$$П=\begin{pmatrix}0.1 & 0 & 0.7\\ 0.1 & 0.3 & 0.2\\ 0.8 & 0.7 & 0.1\end{pmatrix}$$

Численность населения в 2000 году – 474,3 млн. человек.

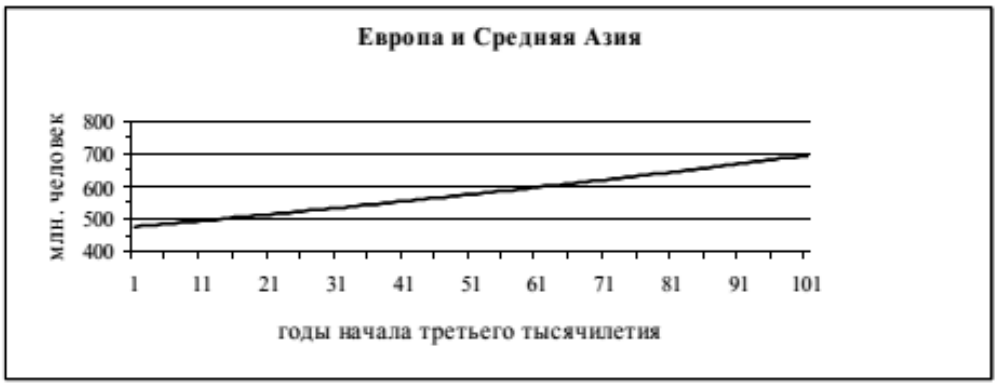


Рис.2. Моделирование динамики изменения численности населения стран Европы и Средней Азии

Разница между прогнозными значениями и фактическими данными в 2002 г. составила около 1,12%. Высокий процент расхождения объясняется, на наш взгляд, высоким уровнем миграции в страны Европы.

Решение:

```
> P := array([[0.1, 0.0, 0.7], [0.1, 0.3, 0.2], [0.8, 0.7, 0.1]]);
```

$$P := \begin{bmatrix} 0.1 & 0. & 0.7 \\ 0.1 & 0.3 & 0.2 \\ 0.8 & 0.7 & 0.1 \end{bmatrix}$$

```
> N := 474300000;
  M := vector([N·22.0/(22 + 12 + 9), N·12/(22 + 12 + 9), N·9/(22 + 12 + 9)]);
```

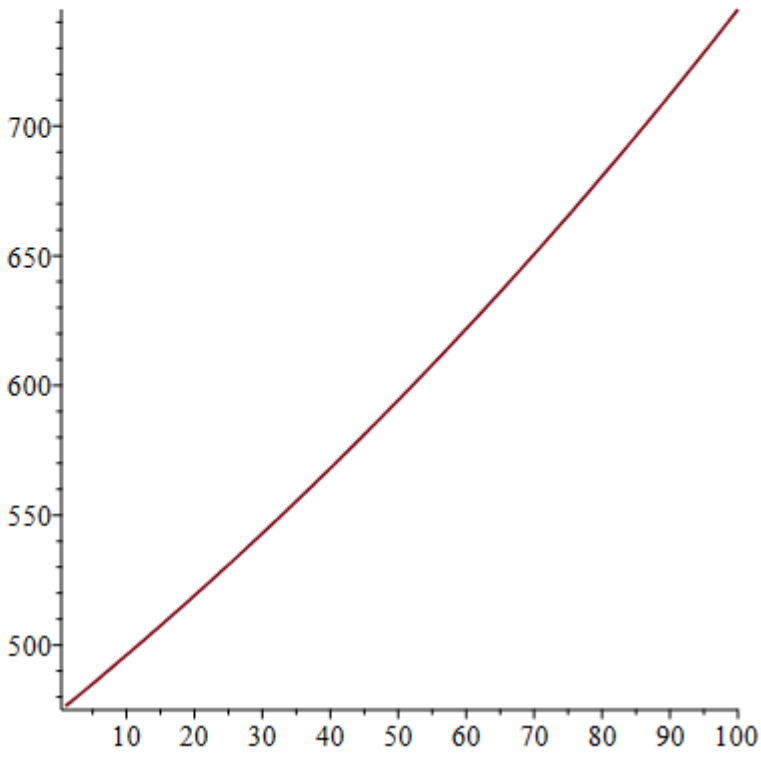
$$N := 474300000$$

$$M := \left[2.426651163\,10^{8} \quad \frac{5691600000}{43} \quad \frac{4268700000}{43} \right]$$

```
> a := vector([(1000 + 22 − 15)/1000, (1000 + 12 − 11)/1000, (1000 + 9 − 5)/1000]);
```

$$a := \left[\frac{1007}{1000} \quad \frac{1001}{1000} \quad \frac{251}{250} \right]$$

```
>
for h to 100 do
Mnew := vector([0, 0, 0]) :
for j to 3 do
for s to 3 do
Mnew[j] := Mnew[j] + P[j, s]·a[s]·M[s] :
end do:
end do:
M := array(Mnew) :
F[h] := (M[1] + M[2] + M[3])/1000000;
end do:
> with(plots) :
> plot({seq([n, F[n]], n = 1 ..100)});
```



Вывод

В ходе лабораторной работы освоили применение метода статистических испытаний к решению практических задач. Была решена задача о поражении цели и получен результат необходимых залпов $n=4$. Так же была решена задача прогнозирования демографической ситуации, прогнозирование изменения численности населения мира в Европе и Средней Азии, полученный график совпадает с ожидаемым результатом.

Приложение

```
#include <iostream>
#include <vector>
using namespace std;

vector<double> TransposeMatrix(vector<double> mat, int n)
{
	vector<double> res;
	for (size_t i = 0; i < n; i++)
	{
		for (size_t j = 0; j < n; j++)
		{
			if (i == j)
			{
				res.push_back(mat[i * n + j]);
				continue;
			}
			else
			{
				res.push_back(mat[j * n + i]);
			}
		}
	}
	return res;
}

double Det(vector<double> m) {
	return m[0] * m[4] * m[8] + m[2] * m[3] * m[7] + m[1] * m[5] * m[6] - m[2] *
m[4] * m[6] - m[0] * m[5] * m[7] - m[1] * m[3] * m[8];
}

vector<double> MatrixMult(vector<double> m1, vector<double> m2)
{
	vector<double> r;
	for (int i = 0; i < 3; i++)
	{
		for (int j = 0; j < 3; j++)
		{
			r.push_back(0.);
			for (int k = 0; k < 3; k++)
				r[3 * i + j] += m1[3 * i + k] * m2[3 * k + j];
		}
	}
	return r;
}

vector<double> MatrixVec(vector<double> mat, vector<double> vec) {
	vector<double> r;
	for (int j = 0; j < 3; j++)
	{
		r.push_back(0.);
		for (int k = 0; k < 3; k++)
			r[j] += mat[3 * j + k] * vec[k];
	}
	return r;
}

vector<double> MatrixPow(vector<double> mat, int n)
{
	vector<double> r;
	if (n == 1) return mat;
	else
	{
		r = mat;
		for (size_t k = 0; k < n; k++)
```

```
			{
				r = MatrixMult(r, mat);
			}
		}
		return r;
}

vector<double> True_P(vector<double> mat) // mat is transpose matrix
{
		vector<double> temp = mat;
		temp[0] -= 1;
		temp[4] -= 1;
		temp[6] = 1;
		temp[7] = 1;
		temp[8] = 1;
		vector<double> det_temp_i;
		for (size_t i = 0; i < 3; i++)
		{
			vector<double> temp_i = temp;
			for (size_t j = 0; j < 2; j++)
			{
				temp_i[j * 3 + i] = 0;
			}
			det_temp_i.push_back(Det(temp_i));
		}
		double det = Det(temp);

		vector<double> res = { det_temp_i[0] / det,det_temp_i[1] / det, det_temp_i[2]
/ det };
		cout << "( " << res[0] << " ; " << res[1] << " ; " << res[2] << " )" << endl;
		return res;

}

bool Check(vector<double> vec1, vector<double> vec2) {
		for (size_t i = 0; i < 3; i++)
		{
			if (abs(vec1[i] - vec2[i]) > 1e-4) return false;
			else { return true; }
		}
}



void PrintMatx(vector<double> mat, int n);

int main()
{
		double a1 = 1. / 14., a2 = (double)rand() / 100., a3 = 1 - a1 - a2;
		vector<double> p0 = { a1,a2,a3 };
		vector<double> p = { 0.5, 0.25, 0.25, 0.5 - 1. / 23., 0., 0.5 + 1. / 23.,
0.25, 0.25, 0.5 };
		vector<double> res_true;
		vector<double> res_iter = { 0.,0.,0. };
		cout << "Original matrix Pi\n";
		PrintMatx(p, 3);

		vector<double> p_T = TransposeMatrix(p, 3);
		cout << "\nTransposed matrix\n";
		PrintMatx(p_T, 3);

		res_true = True_P(p_T);

		bool check = false;
		int n = 0;
		while (check != true) {
			n++;
```

```
            vector<double> temp = MatrixPow(p_T, n);
            res_iter = MatrixVec(temp, p0);
            check = Check(res_true, res_iter);
      }
      cout << "\n( " << res_iter[0] << " ; " << res_iter[1] << " ; " << res_iter[2]
<< " )" << endl;
      cout << "\nn = " << n << endl;
      return 0;
}

void PrintMatx(vector<double> mat, int n)
{
      for (size_t i = 0; i < n; i++)
      {
            for (size_t j = 0; j < n; j++)
            {
                  cout << mat[i * n + j] << "\t";

            }
            cout << "\n";
      }
}
```