Софийски университет „Св. Климент Охридски“

Факултет по математика и информатика

Специалност: “Информационни системи”

Курс: 2, Група: 3

КУРСОВ ПРОЕКТ

по

Небесна механика

Изготвил:

Никола Петров Кирилов(ФН: 71986)

Преподавател:

Доц. Д-р Ангел Живков

София 2021

Задача 1: Пресметнете координатите и скоростите на планетите в деня, в който сте родени.

Легенда:

Орбитата на планетата зависи от 6 елемента (в задачата на Кеплер):

a - дължина на голямата полуос

ecc - екцентрицитет

tilt - наклонение на плоскостта на орбитата

Anomaly - средна аномалия

l0 - средната аномалия в момента **t0**

g + θ - дължина на перихелия

θ – дължина на възела

Пет от тези елементи са константи, единствено средната аномалия **l** е линейна функция на времето **t**.

Допълнителен елемент е ексцентричната аномалия **u**.

Връзката на елиптичните елементи с декартовите координати в $\mathbf{R^3}$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(i) & -\sin(i) \\ 0 & \sin(i) & \cos(i) \end{pmatrix} \times \begin{pmatrix} \cos(g) & -\sin(g) & 0 \\ \sin(g) & \cos(g) & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} z1 \\ z2 \\ 0 \end{pmatrix}$$

Опитах се да направя формулите, които обясняват алгоритъма на Latex, но поради грешка, която ми дава Google Docs, не мога да вмъкна, затова качвам формулите като снимки

Ще взема Кеплеровите елементи и техните стойности от сайта на NASA:
https://ssd.jpl.nasa.gov/txt/aprx_pos_planets.pdf

	a	e	I	L	ϖ	Ω
	$[au, au/cty]$	$[\ ,\ /cty]$	$[deg, deg/cty]$	$[deg, deg/cty]$	$[deg, deg/cty]$	$[deg, deg/cty]$
Mercury	0.38709927	0.20563593	7.00497902	252.25032350	77.45779628	48.33076593
	0.00000037	0.00001906	-0.00594749	149472.67411175	0.16047689	-0.12534081
Venus	0.72333566	0.00677672	3.39467605	181.97909950	131.60246718	76.67984255
	0.00000390	-0.00004107	-0.00078890	58517.81538729	0.00268329	-0.27769418
EM Bary	1.00000261	0.01671123	-0.00001531	100.46457166	102.93768193	0.0
	0.00000562	-0.00004392	-0.01294668	35999.37244981	0.32327364	0.0
Mars	1.52371034	0.09339410	1.84969142	-4.55343205	-23.94362959	49.55953891
	0.00001847	0.00007882	-0.00813131	19140.30268499	0.44441088	-0.29257343
Jupiter	5.20288700	0.04838624	1.30439695	34.39644051	14.72847983	100.47390909
	-0.00011607	-0.00013253	-0.00183714	3034.74612775	0.21252668	0.20469106

Saturn	9.53667594	0.05386179	2.48599187	49.95424423	92.59887831	113.66242448
	-0.00125060	-0.00050991	0.00193609	1222.49362201	-0.41897216	-0.28867794
Uranus	19.18916464	0.04725744	0.77263783	313.23810451	170.95427630	74.01692503
	-0.00196176	-0.00004397	-0.00242939	428.48202785	0.40805281	0.04240589
Neptune	30.06992276	0.00859048	1.77004347	-55.12002969	44.96476227	131.78422574
	0.00026291	0.00005105	0.00035372	218.45945325	-0.32241464	-0.00508664
Pluto	39.48211675	0.24882730	17.14001206	238.92903833	224.06891629	110.30393684
	-0.00031596	0.00005170	0.00004818	145.20780515	-0.04062942	-0.01183482

След това обръщаме **θ , g + θ в Радиани** (*Pi/180)

Обръщаме **i** в градуси (*Pi/180)

Стойностите на **μ**(**маса на планета/маса на слънцето**) са следните:

Планета	**μ**	**Пресмятане на μ (Java дава грешка като го сложа така, затова сложих и пакет BigData, за по-точни сметки)**
Меркурий	1/6023600	1.6601368e-7
Венера	1/408523	0.00000244784
Земя	1/328900,5	0.00000304043
Марс	1/3098708	3.22715145e-7
Юпитер	1/1047,34	0.00095479977
Сатурн	1/3497,8	0.00028589399
Уран	1/22902,9	0.00004366259
Нептун	1/19402	0.00005154107
Плутон	1/135000000	7.40740741e-9

γ= 1 + μ, където **γ =Gm$_A$** $= 6.670 * 10^{-8} \frac{sm^3}{g*sec^2}$ е гравитационна константа.

Величината **n** наричаме **средно движение.**

В сила е и **уравнението на Кеплер: l = u - e.sinu**

Нека **t** – времето от рожденият ми ден до 2000г.

(Рождената ми дата е 18.12.2000г.)

Дните между 01.01.2000 – 18.12.2000 са точно 352, след това намираме t, което е: 352/365.25 => t = **0.9637234770704996,** числото е с 15 знака след запетаята за по-голяма точност, както е по изискване. Сметнах числото на следния онлайн калкулатор за изчисления:

https://www.mathsisfun.com/calculator-precision.html

Решението на задачите ще направя на **Java 15.0.2(SDK Version):**

CelestialMechanicsProblem1.java

```
public class CelestialMechanicsProblem1
{
        //main function to solve the problem
    public void problem1Solver(double a, double ecc, double tilt, double
Anomaly, double w, double Omega, double myu, double t)
    {
        //calculating the theta, g, and the tilt
        double theta = Omega * (Math.PI / 180);
        double g = (w - Omega) * Math.PI / 180;
        tilt = tilt * Math.PI / 180;

        //initializing the thetaMatrix
        double[][] thetaMatrix = {{Math.cos(theta), -Math.sin(theta), 0},
{Math.sin(theta), Math.cos(theta), 0}, {0, 0, 1}};

        //initializing the tiltMatrix
        double[][] tiltMatrix = {{1, 0, 0}, {0, Math.cos(tilt), -
(Math.sin(tilt))}, {0, Math.sin(tilt), Math.cos(tilt)}};

        //initializing the gMatrix
        double[][] gMatrix = {{Math.cos(g), -(Math.sin(g)), 0},
{Math.sin(g), Math.cos(g), 0}, {0, 0, 1}};

        //the q matrix represents the result of the multiplication of the
      //three matrices
              double[][] qMatrix =
      multiplyMatrix(3,3,(multiplyMatrix(3,3,thetaMatrix,3,3,tiltMatrix)),3
      ,3,gMatrix);
              double[][] q1Matrix =
      multiplyMatrix(3,3,(multiplyMatrix(3,3,thetaMatrix,3,3,tiltMatrix)),3
```

```
,3,gMatrix);

        double gama = 1 + myu;
        double n = Math.sqrt(gama/Math.pow(a, 3));
        double to = ((w - Anomaly)/n)*Math.PI/180;
        double l = n*(t*2*Math.PI) + to;
        double u = l + ecc*Math.sin(l + ecc*Math.sin(l +
ecc*Math.sin(l)));

        double[][] rVector = {{Math.cos(u)-ecc},
{Math.sin(u)*Math.sqrt(1-Math.pow(ecc, 2))}, {0}};
        double[][] rResult =
multiplyMatrix(3,3,multiplyMatrixWithConst(q1Matrix,
a,3,3),3,1,rVector);
        double[][] vVector = {{-Math.sin(u)},
{Math.cos(u)*Math.sqrt(1-(ecc*ecc))}, {0}};
                //the multiplication of the q matrix with the vector
         double[][] vResult =
  multiplyMatrix(3,3,qMatrix,3,1,vVector);
         vResult = multiplyMatrixWithConst(vResult,(a*n),3,1);
         vResult = divideMatrixWithConstant(vResult, (1-
  ecc*Math.cos(u)), 3,1);
         System.out.println("R values");
         printVector(rResult);
         System.out.println("V values");
         printVector(vResult);
         System.out.println("|R| value");
         System.out.println(secondNorm(rResult));
         System.out.println("|V| value");
         System.out.println(secondNorm(vResult));

    }

  //method to multiply a matrix with a constant
    private double[][] multiplyMatrixWithConst(double[][] matrix,
  double cosnstant, int rowsNumber, int colsNumber)
    {
        for(int i = 0; i < rowsNumber; i++)
        {
            for (int j = 0; j < colsNumber; j++)
            {
                matrix[i][j] = matrix[i][j]*cosnstant;
            }
        }
        return matrix;
    }
  ////method to divide a matrix with a constant
    private double[][] divideMatrixWithConstant(double[][] matrix,
  double cosnstant, int rowsNumber, int colsNumber)
    {
        for(int i = 0; i < rowsNumber; i++)
        {
            for (int j = 0; j < colsNumber; j++)
            {
                matrix[i][j] = matrix[i][j]/cosnstant;
            }
        }
        return matrix;
    }
```

```
//method to multiply two matrices
 private double[][] multiplyMatrix(int row1, int col1, double
A[][], int row2, int col2, double B[][])
    {
        double[][] tempMatrix = new double[row1][col2];
        for (int i = 0; i < row1; i++) {
            for (int j = 0; j < col2; j++) {
                for (int k = 0; k < row2; k++)
                    tempMatrix[i][j] += A[i][k] * B[k][j];
            }
        }
        return tempMatrix;
    }


       //returning the second normal form of a vector
    double secondNorm(double[][] a)
    {
        double c = 0.0;
        for (int i = 0; i < 3; i++)
        {
                c+= a[i][0] * a[i][0];
        }

        return Math.sqrt(c);
    }

    //method to print a vector
        public void printVector(double[][] vec)
        {
            for (double[] doubles : vec)
            {
                System.out.println(doubles[0]);
            }
        }

    //the main method where we set the NASA data into our method
        public static void main(String[] args)
        {
            CelestialMechanicsProblem1 nebesnaMehanika = new
    CelestialMechanicsProblem1();
            double birthdayToT = (352/365.25);
            System.out.println("Mercury");
            nebesnaMehanika.problem1Solver(0.387, 0.205, 7.004,
    252.250, 77.457, 48.330, 1.6601368e-7, birthdayToT);
            System.out.println("Venus");
            nebesnaMehanika.problem1Solver(0.723, 0.006, 3.394,
    181.979, 131.602, 76.679, 0.00000244784, birthdayToT);
            System.out.println("Earth");
            nebesnaMehanika.problem1Solver(1, 0.016, 0, 100.464,
    102.937, 0,0.00000304043, birthdayToT);
            System.out.println("Mars");
            nebesnaMehanika.problem1Solver(1.523, 0.093, 1.849, -
    4.553, -23.943, 49.559, 3.22715145e-7,birthdayToT);
            System.out.println("Jupiter");
            nebesnaMehanika.problem1Solver(5.202, 0.048, 1.304,
    34.396,14.728, 100.473, 0.00095479977, birthdayToT);
            System.out.println("Saturn");
            nebesnaMehanika.problem1Solver(9.536, 0.053, 2.485,
    49.954, 92.598, 113.662, 0.00028589399, birthdayToT);
            System.out.println("Uranus");
```

```
        nebesnaMehanika.problem1Solver(19.189, 0.047, 0.772,
313.238, 170.954, 74.016, 0.00004366259, birthdayToT);
        System.out.println("Neptune");
        nebesnaMehanika.problem1Solver(30.069, 0.008, 1.770, -
55.120, 44.964, 131.784, 0.00005154107, birthdayToT);
        System.out.println("Pluto");
        nebesnaMehanika.problem1Solver(39.482, 0.248, 17.140,
238.929, 224.068, 110.303, 7.40740741e-9, birthdayToT);

    }
}
```

Получавам следната таблица:

Планета	r	v	|r|	|v|
Меркурий	0.3190 0.1034 -0.0208	-0.8203 1.6287 0.2083	0.33598	1.8354
Венера	0.4169 -0.5951 -0.0322	0.9561 0.6712 -0.0460	0.72731	1.1691
Земя	-0.0345 0.9837 0	-1.0151 -0.0386 0	0.98428	1.0158
Марс	-1.0426 1.2734 0.0523	-0.5989 -0.4465 0.0054	1.6466	0.74702
Юпитер	-5.3885 0.6754 0.1178	-0.0599 -0.4154 0.0031	5.432	0.41967
Сатурн	1.6127 -9.9066 0.1084	0.3027 0.0511 -0.0129	10.0376	0.30726

Уран	0.0135 19.0066 0.0703	-0.2302 -0.0105 0.0029	19.0068	0.23047
Нептун	29.2893 -6.1648 -0.5480	0.0364 0.1795 -0.0045	29.9361	0.18318
Плутон	-14.5641 38.4122 0.1020	-0.1191 -0.0853 0.0436	41.0806	0.15283

Задача 2: Пресметнете елементите на Делоне и Поанкаре (от първи и втори вид) в деня, в който сте родени

Легенда:

Елементите на Делоне **– L, G, Θ, l, g, θ** се изразяват чрез орбиталните елементи:

- **a** - дължина на голямата полуос,
- **ecc** - екцентрицитет,
- **tilt** - наклонение на плоскостта на орбитата,
- **Anomaly**- средна аномалия, (l0 е средната аномалия в момента t0),
- **g + θ** - дължина на перихелия,
- **θ** – дължина на възела.
- **(l,L), (G,g) и (Θ, θ**) са спрегнати канонични променливи

И **t** от предната задача **t = 0.9637234770704996 * 2pi => t = 6.055253191313386**

Поанкаре дефинира две системи от по шест елемента, характеризиращи орбитите на планетите:

Първа система от шест елемента, характеризираща орбитите на планетите:

$$\begin{pmatrix} L & L - G & G - \Theta \\ l + g + \theta & -g - \theta & -\theta \end{pmatrix}$$

И втората:

$$\begin{pmatrix} L & \xi := \sqrt{2(L-G)}\cos(g+\theta) & p := \sqrt{2(G-\Theta)}\cos(\theta) \\ \lambda := l + g + \theta & \eta := -\sqrt{2(L-G)}\sin(g+\theta) & q := -\sqrt{2(G-\Theta)}\sin(\theta) \end{pmatrix}$$

Елементи на **Делоне** показани в следната таблица(Резултат от изпълнението на програмата)**:**

Код за задачата:

CelestialMechanicsProblem2.java

```
public class CelestialMechanicsProblem2
{
        //the main method where we set the NASA data into our method
    public void problem2Solver( double a, double ecc, double i, double
Anomaly, double w, double Omega, double myu, double t)
    {
        i = i * Math.PI/180;
        double n = Math.sqrt(1 / (a*a*a));
        double to = ((w - Anomaly) / n) * Math.PI/180;

        double gamma = 1 + myu;
        double L = (myu) * Math.sqrt(gamma*a);
        double G = L * Math.sqrt(1 - ecc*ecc);
        double bigTheta = G*Math.cos(i);

        double l = n * (t*2*Math.PI) + to;
        double g = (w - Omega) * Math.PI/180;
        double theta = Omega * Math.PI/180;
        double H = -myu*gamma / (2*a);

        double FirstPoincare11 = L;
        double FirstPoincare12 = L - G;
        double FirstPoincare13 = G - bigTheta;
        double FirstPoincare21 = l + g + theta;
        double FirstPoincare22 = -g - theta;
        double FirstPoincare23 = -theta;
        double SecondPoincare11 = FirstPoincare11;
        double SecondPoincare12 = Math.sqrt(2 * (L - G)) * Math.cos(g +
theta);
        double SecondPoincare13 = Math.sqrt(2 * (G - bigTheta)) *
Math.cos(theta);
        double SecondPoincare21 = FirstPoincare21;
        double SecondPoincare22 = -Math.sqrt(2 * (L - G)) * Math.sin(g +
theta);
        double SecondPoincare23 = -Math.sqrt(2 * (G - bigTheta)) *
Math.sin(theta);

        System.out.println("L: "+ L);
```

```
        System.out.println("G: "+ G);
        System.out.println("ThetaBig: "+ bigTheta);
        System.out.println("l: "+ l);
        System.out.println("g: "+ g);
        System.out.println("thetaSmall: "+ theta);
        System.out.println("H:" + H);
        System.out.println("FirstPoincare11: "+ FirstPoincare11);
        System.out.println("FirstPoincare12: "+ FirstPoincare12);
        System.out.println("FirstPoincare13: "+ FirstPoincare13);
        System.out.println("FirstPoincare21: "+ FirstPoincare21);
        System.out.println("FirstPoincare22: "+ FirstPoincare22);
        System.out.println("FirstPoincare23: "+ FirstPoincare23);
        System.out.println("SecondPoincare11: "+ SecondPoincare11);
        System.out.println("SecondPoincare12: "+ SecondPoincare12);
        System.out.println("SecondPoincare13: "+ SecondPoincare12);
        System.out.println("SecondPoincare21: "+ SecondPoincare21);
        System.out.println("SecondPoincare22: "+ SecondPoincare22);
        System.out.println("SecondPoincare23: "+ SecondPoincare23);

    }

        //the main method where we
    public static void main(String[] args)
    {
        CelestialMechanicsProblem2 nebesnaMehanika = new
CelestialMechanicsProblem2();
        double birthdayToT = (352/365.25);
        System.out.println("Mercury");
        nebesnaMehanika.problem2Solver(0.387, 0.205, 7.004, 252.250,
77.457, 48.330, 1.6601368e-7, birthdayToT);
        System.out.println("Venus");
        nebesnaMehanika.problem2Solver(0.723, 0.006, 3.394, 181.979,
131.602, 76.679, 0.00000244784, birthdayToT);
        System.out.println("Earth");
        nebesnaMehanika.problem2Solver(1, 0.016, 0, 100.464, 102.937,
0,0.00000304043, birthdayToT);
        System.out.println("Mars");
        nebesnaMehanika.problem2Solver(1.523, 0.093, 1.849, -4.553, -
23.943, 49.559, 3.22715145e-7,birthdayToT);
        System.out.println("Jupiter");
        nebesnaMehanika.problem2Solver(5.202, 0.048, 1.304, 34.396,14.728,
100.473, 0.00095479977, birthdayToT);
        System.out.println("Saturn");
        nebesnaMehanika.problem2Solver(9.536, 0.053, 2.485, 49.954, 92.598,
113.662, 0.00028589399, birthdayToT);
        System.out.println("Uranus");
        nebesnaMehanika.problem2Solver(19.189, 0.047, 0.772, 313.238,
170.954, 74.016, 0.00004366259, birthdayToT);
        System.out.println("Neptune");
        nebesnaMehanika.problem2Solver(30.069, 0.008, 1.770, -55.120,
44.964, 131.784, 0.00005154107, birthdayToT);
        System.out.println("Pluto");
        nebesnaMehanika.problem2Solver(39.482, 0.248, 17.140, 238.929,
224.068, 110.303, 7.40740741e-9, birthdayToT);

    }
}
```

	L	**G**	**Θ**	**l**	**g**	**θ**	**H**
Меркурий	1.0328e-07	1.0108e-07	1.0033e-07	24.4172	0.5084	0.8435	-2.1449e-07
Венера	2.0814e-06	2.0814e-06	2.0777e-06	9.3092	0.9586	1.3383	-1.6928e-06
Земя	3.0404e-06	3.0400e-06	3.0400e-06	6.0984	1.7966	0	-1.5202e-06
Марс	3.9826e-07	3.9654e-07	3.9633e-07	2.5856	-1.2829	0.8650	-1.0595e-07
Юпитер	0.0022	0.0022	0.0022	-3.5624	-1.4965	1.7536	-9.1860e-05
Сатурн	8.8298e-04	8.8174e-04	8.8091e-04	22.1228	-0.3676	1.9838	-1.4995e-05
Уран	1.9127e-04	1.9106e-04	1.9104e-04	-208.6710	1.6919	1.2918	-1.1377e-06
Нептун	2.8263e-04	2.8262e-04	2.8249e-04	288.0553	-1.5153	2.3001	-8.5709e-07
Плутон	4.6544e-08	4.5090e-08	4.3088e-08	-64.3220	1.9856	1.9252	-9.3807e-11

Сега нека видим **първа система на Поанкаре**

Планета	**L**	**L-G**	**G-Θ**	**l + g + θ**	**-g - θ**	**-θ**
Меркурий	1.0328e-07	2.1934e-09	7.5431e-10	25.7690	-1.3519	-0.8435
Венера	2.0814e-06	3.7465e-11	3.6506e-09	11.6061	-2.2969	-1.3383
Земя	3.0404e-06	3.8920e-10	0	7.8950	-1.7966	0
Марс	3.9826e-07	1.7260e-09	2.0646e-10	2.1677	0.4179	-0.8650
Юпитер	0.0022	2.5114e-06	5.6359e-07	-3.3054	-0.2571	-1.7536
Сатурн	8.8298e-04	1.2410e-06	8.2918e-07	23.7389	-1.6161	-1.9838
Уран	1.9127e-04	2.1137e-07	1.7343e-08	-205.6873	-2.9837	-1.2918
Нептун	2.8263e-04	9.0444e-09	1.3485e-07	288.8401	-0.7848	-2.3001
Плутон	4.6544e-08	1.4540e-09	2.0026e-09	-60.4112	-3.9107	-1.9252

Сега нека видим **втора система на Поанкаре**

Планета	**L**	**ξ**	**p**	**λ = l + g + θ**	**η**	**q**
Меркурий	1.0328e-07	1.4384e-05	2.5823e-05	25.7690	-6.4652e-05	-2.9014e-05
Венера	2.0814e-06	-5.7473e-06	1.9688e-05	11.6061	-6.4729e-06	-8.3148e-05
Земя	3.0404e-06	-6.2462e-06	0	7.8950	-2.7192e-05	0
Марс	3.9826e-07	5.3698e-05	1.3181e-05	2.1677	2.3844e-05	-1.5466e-05
Юпитер	0.0022	0.0022	-1.9299e-04	-3.3054	-5.6977e-04	-0.0010

Сатурн	8.8298e-04	-7.1412e-05	-5.1684e-04	23.7389	-0.0016	-0.0012
Уран	1.9127e-04	-6.4210e-04	5.1285e-05	-205.6873	-1.0223e-04	-1.7904e-04
Нептун	2.8263e-04	9.5162e-05	-3.4604e-04	288.8401	-9.5042e-05	-3.8724e-04
Плутон	4.6544e-08	-3.8747e-05	-2.1959e-05	-60.4112	3.7507e-05	-5.9354e-05

Използвани материали:

https://www.fmi.uni-sofia.bg/sites/default/files/users/u788/nebesna_mehanika.rar