Übungsblatt 4: Das Torkeln des Marsmondes Phobos

Wayne Ströbel und Silas Müller

21.6.2021

Aufgabenstellung

Der Marssatellit Phobos umkreist den Mars auf einer festen elliptischen Bahn mit Radius $r(t)$ und Polarwinkel $\phi(t)$ mit großer Halbachse a und Exzentrizität ϵ. Zusätzlich führt Phobos noch eine Eigenrotation in der Ebenen der Bewegung durch, die durch den Winkel $\vartheta(t)$ beschrieben wird. Der Satellit hat eine stark unsymmetrische Form, so dass seine drei Trägheitsmomente wie folgt betraglich geordnet sind $I_1 < I_2 < I_3$ Da die Ellipse fest ist, kann die Bewegungsgleichung für $\vartheta(t)$ wie folgt geschrieben werden (siehe Vorlesung):

$$I_3\ddot{\vartheta}(t) = -\frac{3}{2}(\frac{2\pi}{T})^2(I_2 - I_1)(\frac{a}{r(t)})^3 \sin(2(\vartheta(t) - \phi(t))$$

Dabei bezeichnet T die volle Umlaufzeit des Satelliten. Diese Gleichung hat formal nur einen Freiheitsgrad, über r und ϕ gibt es aber eine implizite Zeitabhängigkeit.
Dazu wird zunächst die DGL genauer betrachtet, sowie eine Beziehung zwischen r und ϕ, sowie eine DGL für ϕ hergeleitet.
Abschließend werden die Ergebnisse in einer Poincaré Abbildung dargestellt, in der wir $\frac{T\dot{\vartheta}(t)}{2\pi}$ gegen $\vartheta/)$ am Perihel der Satellitenbahn auftragen.Insbesondere sollen chaotische und (quasi-)periodische Bahnkurven miteinander verglichen werden.

Weitere Beziehungen und Differentialgleichungen

Im folgenden gilt:

$$\alpha^2 = 3\frac{I_2 - I_1}{I_3} = 0.83^2, \;\; \epsilon = 0.015$$

Wandelt man die Ellipsengleichung in Polarkoordinaten um, vereinfacht und löst nach r auf, so ergibt sich mit $p = \frac{b^2}{a}$, wobei a die große und b die kleine Halbachse der Ellipse darstellt:

$$r = \frac{p}{1 + \epsilon cos(\phi)}$$

Des Weiteren gilt:

$$\epsilon = \frac{\sqrt{a^2 - b^2}}{a} \;\; also: \;\; p = a(1 - \epsilon^2)$$

Schreibt man nun r in Einheiten von a um: $\tilde{r} = \frac{r}{a}$

$$\tilde{r}(t) = \frac{1 - \epsilon^2}{1 + \epsilon cos\phi(t)}$$

Wir wollen nun noch eine DGL für ϕ bestimmen. Dazu betrachten wir die Drehimpulserhaltung:

$$\dot{L} = m\dot{\omega}r^2 + 2m\omega r\dot{r} = 0$$

$$\Rightarrow m\ddot{\phi}\frac{p^2}{(1+\epsilon cos(\phi))^2} + 2m\dot{\phi}\frac{p^2 sin(\phi)\epsilon}{(1+\epsilon cos(\phi))^3}\dot{\phi} = 0$$

$$\Rightarrow 0 = \ddot{\phi}(t) + 2\dot{\phi}^2(t)\epsilon\frac{sin(\phi)}{1+\epsilon cos(\phi)}$$

Führt man jetzt noch eine Zeittransformation: $\tau = \frac{t}{T}$ ein, so gilt:

$$\tilde{r}(\tau) = \frac{1-\epsilon^2}{1+\epsilon cos(\phi(\tau))}$$

$$\ddot{\phi}(\tau) = -2\dot{\phi}^2(\tau)\epsilon\frac{sin(\phi(\tau))}{1+\epsilon cos(\phi(\tau))}$$

und abschließend:

$$\ddot{\vartheta}(\tau) = -\frac{\pi^2}{2}\alpha^2(\tilde{r}(\tau))^{-3}\sin(2(\vartheta(\tau)-\phi(\tau))$$

Numerische Lösung der DGLs

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <algorithm>
#include <Rcpp.h>

using namespace Rcpp;

typedef struct
{
    double R,Phi,Phi_pri,Teta,Teta_pri;   //pri steht für Ableitung
} Status;

typedef struct
{
    double eps, T, a, alpha, LM;
} Parameter;

//Definitionen für f-Funktionen zum Integrieren

void f(Status oldStatus, Parameter parameter, Status& newStatus){

    newStatus.Phi_pri = - 2 * parameter.eps * sin(oldStatus.Phi) /
    (1 + parameter.eps * cos(oldStatus.Phi)) * pow(oldStatus.Phi_pri, 2);
    newStatus.Phi = oldStatus.Phi_pri;
    newStatus.R = parameter.T * (1. - (parameter.eps * parameter.eps)) *
    parameter.eps * sin(oldStatus.Phi) / ( (1. + parameter.eps * cos(oldStatus.Phi)) *
    (1. + parameter.eps * cos(oldStatus.Phi))) * oldStatus.Phi_pri;
    newStatus.Teta_pri = -1./2. * pow(2.* M_PI, 2) * pow(parameter.alpha, 2) *
    pow(1/oldStatus.R, 3) * sin(2.*(oldStatus.Teta - oldStatus.Phi));
    newStatus.Teta = oldStatus.Teta_pri;

    // printf("%.10f %.10f %.10f %.10f \n", newStatus.Phi, newStatus.R, newStatus.Teta_pri, newStatus.T
}
```

```
//RungeKutta Schritt

void RungeKutta(Status oldStatus, Parameter parameter, Status& newStatus, double h) {

    Status f_i;
    f(oldStatus, parameter, f_i);

    //Definition für f(x_(i+1))
    Status f_i1;

    //Beschreibung zur Berechnung von f_i1
    Status rechnung;
    rechnung.Phi_pri = oldStatus.Phi_pri + h*f_i.Phi_pri;
    rechnung.Phi = oldStatus.Phi + h*f_i.Phi;
    rechnung.R = oldStatus.R + h*f_i.R;
    rechnung.Teta_pri = oldStatus.Teta_pri + h*f_i.Teta_pri;
    rechnung.Teta = oldStatus.Teta + h*f_i.Teta;

    f(rechnung, parameter, f_i1);

    //Rechenschritt
    newStatus.Phi_pri = oldStatus.Phi_pri + h/2. *(f_i.Phi_pri + f_i1.Phi_pri);
    newStatus.Phi = oldStatus.Phi + h/2. * (f_i.Phi + f_i1.Phi);
    newStatus.R = oldStatus.R + h/2. * (f_i.R + f_i1.R);
    newStatus.Teta_pri = oldStatus.Teta_pri + h/2. * (f_i.Teta_pri + f_i1.Teta_pri);
    newStatus.Teta = oldStatus.Teta + h/2. * (f_i.Teta + f_i1.Teta);

}

//[[Rcpp::export]]
List Simulation(double a, double alpha, double T, double eps, double LM,
                double Phi0, double Phi_pri0, double R0, double Teta0, double Teta_pri0,
                double h, double Rounds){

    NumericVector Teta(0);
    NumericVector Teta_Pri(0);

    //Definition der Parameter
    Parameter phobos;
    phobos.a = a;
    phobos.alpha = alpha;
    phobos.T = T;
    phobos.eps = eps;
    phobos.LM = LM;

    //Anfangswerte
    Status old;
    old.Phi = Phi0;
    old.R = R0 - pow(phobos.eps, 2);
    old.Teta = Teta0;
    old.Teta_pri = Teta_pri0;
    old.Phi_pri = Phi_pri0;
```

```
    int k = 0;
    for (double i = 0; i < Rounds ; i+=h)
    {
        RungeKutta(old, phobos, old, h);

    if ((k+1)*M_PI*2 < old.Phi)   //Prüft auf vielfache von 2*Pi von Phi
    {

        Teta.push_back(old.Teta);
        Teta_Pri.push_back(old.Teta_pri*phobos.T / (2*M_PI));


        k+=1;
    }
    }
    List L = List::create(Teta, Teta_Pri);

    return L;
}
```

Poincare-Abbildungen für verschiedene Anfangswerte

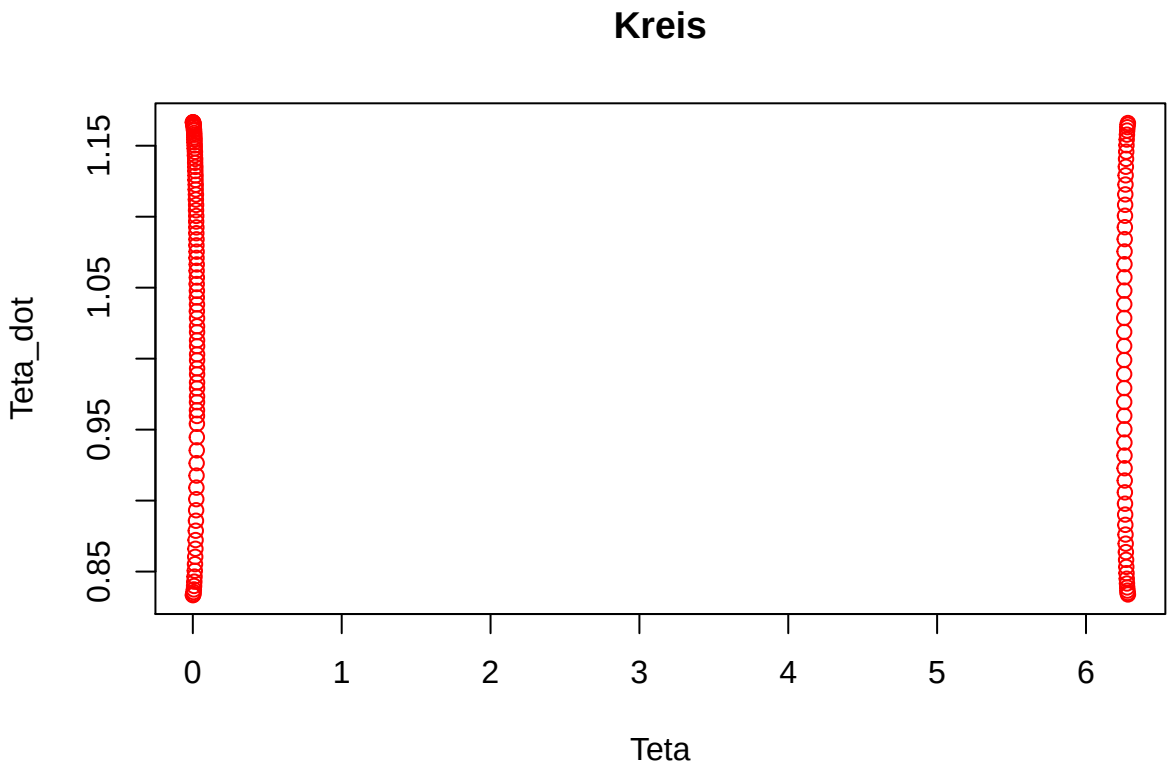
Es werden nun die oben beschriebenen Poincare-Abbildungen für verschiedene Anfangswerte dargestellt. Bei allen Abbildungen wird $\vartheta \mod 2\pi$ aufgetragen.

Für ϵ=0

Bei $\epsilon = 0$ handelt es sich um eine Kreisbahn (also konstantes $r(t)$) die Phobos um den Mars absolviert. Die Eigenrotation ist periodisch, also insbesondere nicht chaotisch.

```
#Parameter und Anfangswerte
a <- 1.
alpha <- 0.82
T_um <- 7.33
eps <- 0.
LM <- 1.

h <- 0.001
Rounds <- 1000

R0 <- 1.
Phi0 <- 0.
Teta0 <- 0.
Phi_Pri0 <- 2*pi/T_um
Teta_Pri0 <- 1.

List <- Simulation(a, alpha, T_um, eps, LM, Phi0, Phi_Pri0, R0, Teta0, Teta_Pri0, h, Rounds)
Teta <- unlist(List[1])
Teta_Pri <- unlist(List[2])

plot(Teta%%(2*pi), Teta_Pri,
     main="Kreis",
```

```
     xlab="Teta",
     ylab="Teta_dot",
     col="red",
     type="p")
```

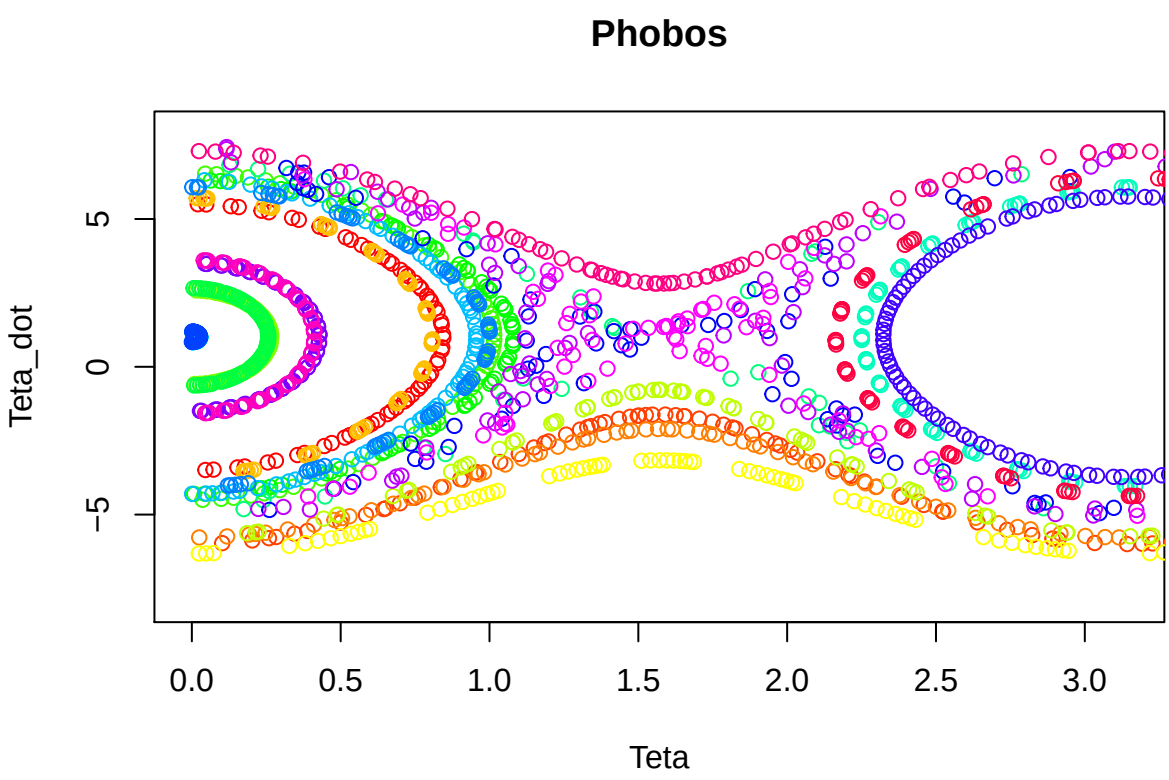


verschiedene Anfangswerte für Phobos

Wir wählen nun die realen Daten mit $\epsilon = 0.015$. Phobos befindet sich nun also auf einer Ellipse um den Mars.

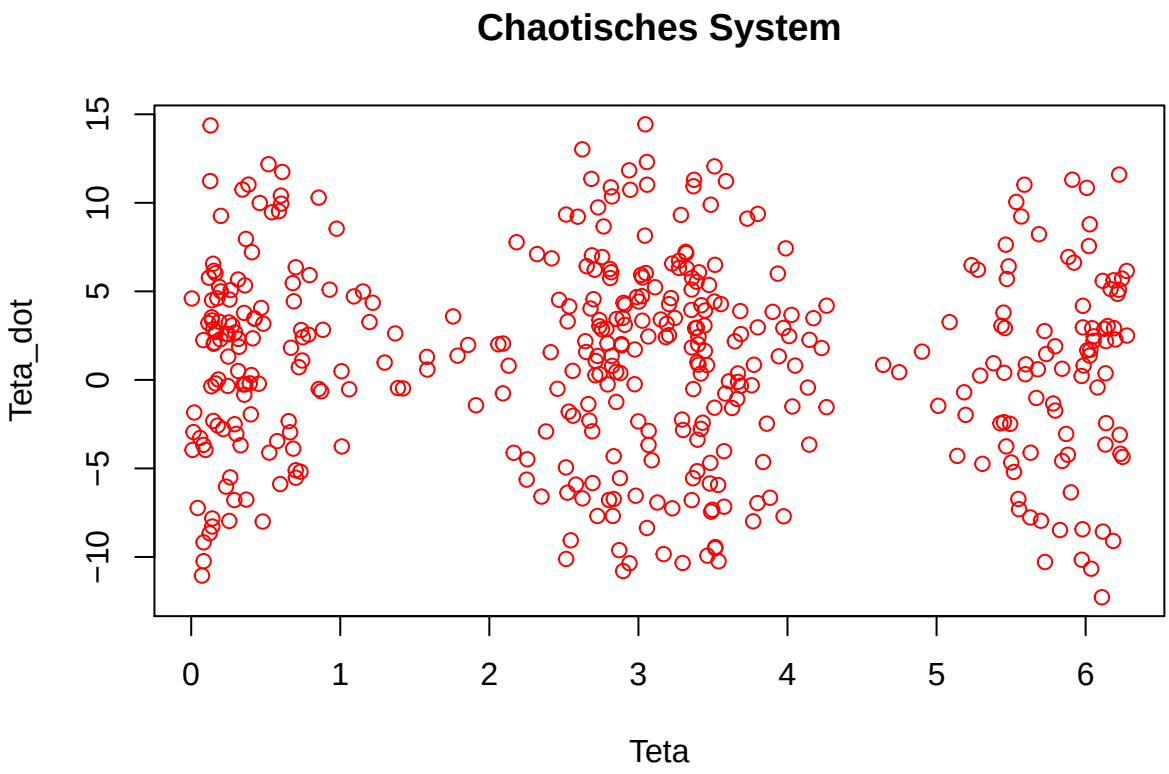
```
#Parameter und Anfangswerte
a <- 1.
alpha <- 0.82
T_um <- 7.33
eps <- 0.015
LM <- 1.

h <- 0.001
Rounds <- 1000

R0 <- 1.
Phi0 <- c(0,1)
Teta0 <- c(0,1,5)
Phi_Pri0 <- 2*pi/T_um
```

```
Teta_Pri0 <- c(-3,-0.5, 1,3)

for (i in Teta_Pri0) {
  for (j in Teta0) {
    for (k in Phi0){

      name <- paste('',k,j,i,sep='')

      List[[name]] <- Simulation(a, alpha, T_um, eps, LM, k, Phi_Pri0, R0, j, i, h, Rounds)

    }
  }
}

var <- 1
cols <- rainbow(length(Phi0)*length(Teta0)*length(Teta_Pri0))

plot(0, 0,
     main="Phobos",
     xlim=c(0,pi),
     ylim=c(-8,8),
     xlab="Teta",
     ylab="Teta_dot",
     type="n")

for (i in Teta_Pri0) {
  for (j in Teta0) {
    for (k in Phi0){

      name <- paste('',k,j,i,sep='')

      points(unlist(List[[name]][1])%%(2*pi), unlist(List[[name]][2]), col=(cols[var]))
      var <- var + 1

    }
  }
}
```

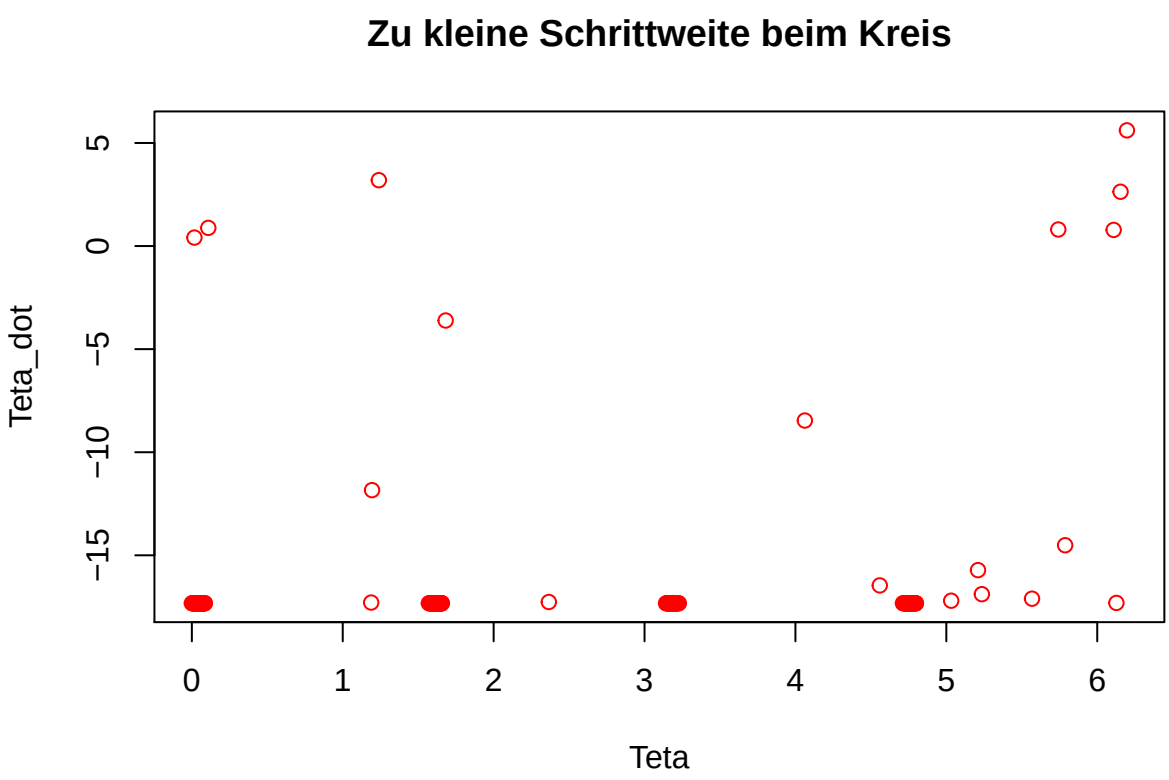


Diskussion

Wie man in den obigen Abbildungen erkennt, varriert die Eigenrotation von Phobos sehr stark bei einer geringen Veränderung der Anfangswerte. So lassen sich gut quasi periodische Eigenrotationen aber auch stark chaotische erkennen.



Die Stabilität des Verfahrens ist nur bei genügend kleiner Schrittweite (bei uns $h = 0.001$) gut gegeben. Ist dies nicht der Fall wird das Verfahren sehr schnell instabil. Was man im folgenden sieht:



Fazit

Wir konnten die gefordeten Tranformationen der DGL durchführen, sowie mittels Drehimpulserhaltung und den Ellipsengeleichungen einen Zusammenhang zwischen $r(t)$ und $\phi(t)$ herstellen und so eine DGL für $\phi(\tau)$ herleiten. Mit Hilfe dieser konnten wir das vollständige Systme der DGLs aufstellen und diese mittels Runge-Kutta lösen.
Wie in der Vorlesung angedeutet, handelt es sich für bestimmte Anfangswerte um ein chaotisches System. Die Eigenrotation des Phobos kann daher nur für sehr bestimmte Anfangswerte periodisch beobachtet werden. Die Stabilität unserer Verfahrens hängt stark von der Schrittweite ab. Für genügend kleine Schrittweite ist dies aber gewährleistet.

Literatur

Vorlesungskript, sowie Vorlesung vom 9.6.2021