Übung 04: Das Torkeln des Marsmondes Phobos

Tobias Blesgen und Leonardo Thome

23.06.2021

Im Folgenden wollen wir das Torkeln des Marsmondes Phobos untersuchen, welches durch die Ellipsoideform des Modes und der elliptischen Umlaufbahn um den Mars entsteht. \ Dazu beschreiben wir die Situation wie folgt:

Wir nehmen an, dass der Mond Phobos den Mars auf einer festen elliptischen Bahn mit Radius $r(t)$, Polarwinkel $\phi(t)$, großen Halbachse a und Exzentrizität ϵ in der Umlaufzeit T umkreist. Die Eigenbewegung von Phobos wird durch den Winkel $\theta(t)$ beschrieben, mit den drei Trägheitsmomenten $I_1 < I_2 < I_3$ von Phobos, kann die Bewegungsgleichung für die Eigenbewegung wie folgt beschrieben werden:

$$I_3\ddot{\theta}(t) = -\frac{3}{2}(\frac{2\pi}{T})^2(I_2 - I_1)(\frac{a}{r(t)})^3 \sin 2(\theta(t) - \phi(t)) \tag{1}$$

Mit der Größe $\alpha = \sqrt{3\frac{I_2 - I_1}{I_3}}$ vereinfachen wir die Gleichung zu:

$$\ddot{\theta}(t) = -\frac{\alpha^2}{2}(\frac{2\pi}{T})^2(\frac{a}{r(t)})^3 \sin 2(\theta(t) - \phi(t)) \tag{2}$$

Dabei lässt sich die Differentialgleichung 2.Ordnung auch als 2 Differentialgleichungen 1.Ordnung schreiben:

$$\frac{\mathrm{d}\theta(t)}{\mathrm{d}t} = \dot{\theta} \tag{3}$$

$$\frac{\mathrm{d}\dot{\theta}(t)}{\mathrm{d}t} = -\frac{\alpha^2}{2}(\frac{2\pi}{T})^2(\frac{a}{r(t)})^3 \sin 2(\theta(t) - \phi(t)) \tag{4}$$

Da $r(t)$ und $\phi(t)$ selbst zeitabhängig sind, müssen wir erst diese lösen, um die Lösen zu $\theta(t)$ finden zu können.

Nach den Keplerschen Gesetzen erhalten wir die Beziehung (QUELLE):

$$r(\phi) = \frac{p}{1 + \epsilon \cos\phi} = \frac{a(1 - \epsilon^2)}{1 + \epsilon\cos\phi} \tag{5}$$

und für die Ableitung nach dem Winkel:

$$\frac{\partial r(\phi)}{\partial\phi} = \frac{a(1 - \epsilon^2)\epsilon\sin\phi}{(1 + \epsilon\cos\phi)^2} \tag{6}$$

Aus der Drehimpulserhaltung können wir folgern(QUELLE):

$$L = mr^2 * \dot{\phi} = const \leftrightarrow \dot{\phi} \qquad = \frac{L}{M}\frac{1}{r^2} \tag{7}$$

Für die zeitliche Ableitung von r ergibt sich mithilfe der Gleichungen (6) & (8):

$$\dot{r} = \frac{\partial r}{\partial\phi}\frac{\partial\phi}{\partial t} = \frac{a(1 - \epsilon^2)\epsilon\sin\phi}{(1 + \epsilon\cos\phi)^2}\frac{L}{M}\frac{1}{r^2} \tag{8}$$

Mit den 4 Differntialgleichung können wir nun $\theta(t)$ und $\dot{\theta}(t)$ bestimmen (r ist dabei in Einheiten von a).

$$\frac{\mathrm{d}\phi}{\mathrm{d}t} = \frac{L}{M}\frac{1}{r^2} \tag{9}$$

$$\frac{\mathrm{d}r}{\mathrm{d}t} = \frac{a(1-\epsilon^2)\epsilon \sin\phi}{(1+\epsilon\cos\phi)^2}\frac{L}{M}\frac{1}{r^2} \tag{10}$$

$$\frac{\mathrm{d}\theta(t)}{\mathrm{d}t} = \dot{\theta} \tag{11}$$

$$\frac{\mathrm{d}\dot{\theta}(t)}{\mathrm{d}t} = -\frac{\alpha^2}{2}(\frac{2\pi}{T})^2(\frac{a}{r(t)})^3 \sin 2(\theta(t) - \phi(t)) \tag{12}$$

Runge-Kutta 2 Verfahren

Um das Differentialgleichungssysteme auszuwerten, verwenden wir das Runge-Kutta Verfahren nach

$$x_{i+1} = x_i + \frac{h}{2}[f(t_i, x_i) + f(t_i + h, x_i + hf(t_i, x_i))]. \tag{13}$$

Wobei sich unser f aus den vier Anteilen von ϕ, r, θ und $\dot{\theta}$ zusammensetzt.

Dabei bietet uns das Runge-Kutter Verfahren numerische Stabilität (soweit das System selbst Stabile ist) und weist mit einem Verfahrensfehler von $\mathcal{O}(h^2)$ einen kleineren Fehler als das Eulerverfahren auf bei keinen Schrittweiten h.

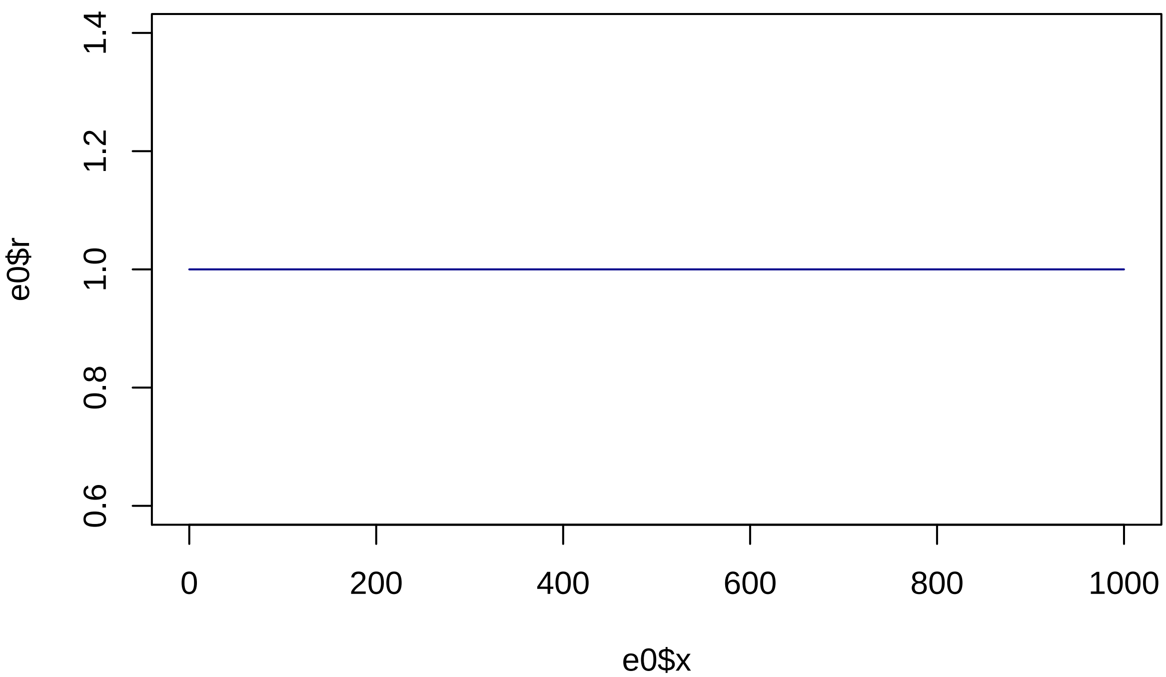
Implementation des DGS nach dem Runge-Kutta 2 Verfahren

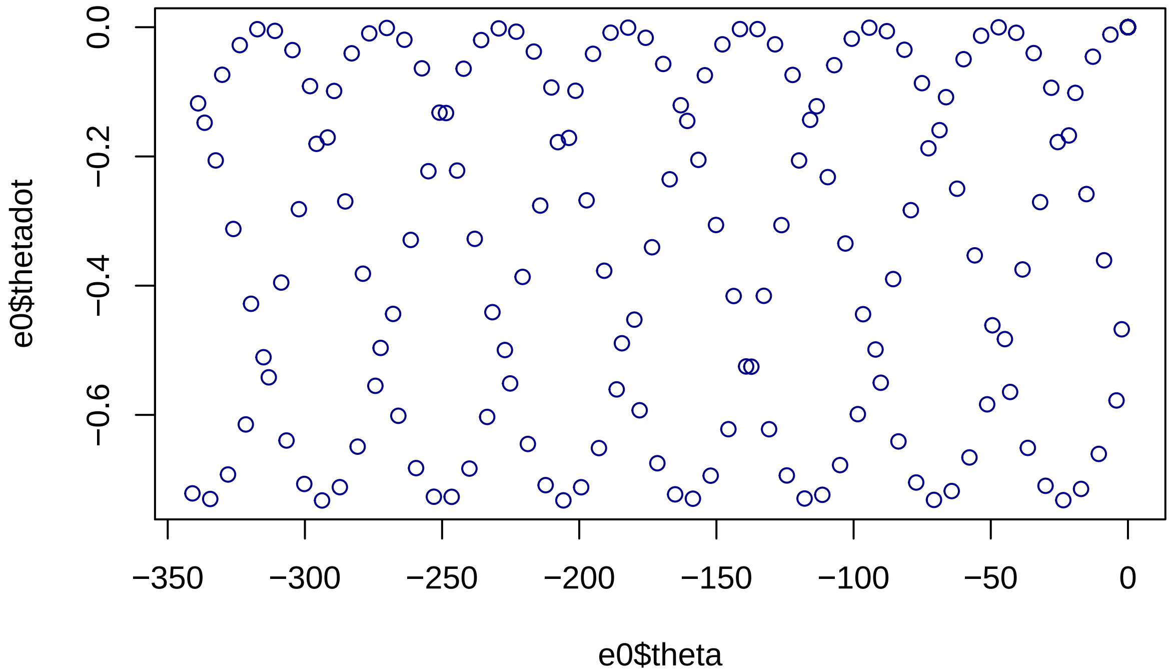
```
#include<Rcpp.h>
#include<stdlib.h>
#include<vector>
#include<algorithm>

using namespace Rcpp;

// Wir verwenden Strukturen, um Funktionsargumente uebersichtlicher zu halten
typedef struct
{
    double phi, r, theta,thetadot;
} Status;

typedef struct
{
    double epsilon,LM,ktheta;
} Parameter;

// Template zum Zerschneiden der verwendeten Vektoren
template<typename T>
std::vector<T> slice(std::vector<T> const &v, int m, int n)
{
    auto erste = v.cbegin() + m;
```

```
    auto letzte = v.cbegin() + n + 1;

    std::vector<T> vektor(erste, letzte);
    return vektor;
}

// Berechnungsschritt der Ableitungen nach dem DGS
void f(Status alterStatus, Parameter parameter, Status& neuerStatus){

    neuerStatus.r = parameter.LM*(1-parameter.epsilon*parameter.epsilon)*parameter.epsilon*sin(alterSta

    neuerStatus.phi = parameter.LM/(alterStatus.r*alterStatus.r);

    neuerStatus.theta = alterStatus.thetadot;

    neuerStatus.thetadot = parameter.ktheta*sin(2*(alterStatus.theta-alterStatus.phi))/(alterStatus.r*al


    //neuerStatus.phi = parameter.LM*(1+2*parameter.epsilon*cos(alterStatus.phi)+parameter.epsilon*para

}

// Ein Intergrationsschritt nach Runge-Kutta
void rkSchritt(Status& status, Parameter parameter, double h){
    Status fStatus;                 // Standart Ableitung
    f(status, parameter, fStatus);

    Status f2Status;                // Mischterm Ableitung
    Status gemischt = {.phi = status.phi + h*fStatus.phi,.r = status.r + h*fStatus.r,
                       .theta = status.theta + h*fStatus.theta,.thetadot = status.thetadot + h*fStatus.t

    f(gemischt, parameter, f2Status);
    status.phi = status.phi + h/2*(fStatus.phi + f2Status.phi);
    status.r = status.r+ h/2*(fStatus.r + f2Status.r);
    status.theta = status.theta + h/2*(fStatus.theta + f2Status.theta);
    status.thetadot = status.thetadot + h/2*(fStatus.thetadot + f2Status.thetadot);
    }

//[[Rcpp::export]]
Rcpp::List durchlauf(const int maxSchritte, const double h,
                            const double phi, const double r, const double theta,
                            const double thetadot,const double epsilon,const double LM,
                            const double ktheta,const double x0){
  // Arrays der Werte zur späteren Ausgabe
    std::vector<double> xWerte(maxSchritte);
    std::vector<double> phiWerte(maxSchritte);
    std::vector<double> rWerte(maxSchritte);
    std::vector<double> thetaWerte(maxSchritte);
    std::vector<double> thetadotWerte(maxSchritte);
    int k = 0;
  // Quelltext
  Status status = {.phi = phi, .r = r, .theta = theta,.thetadot = thetadot};
  Parameter parameter = {.epsilon = epsilon,.LM = LM, .ktheta = ktheta};
```

```
  // Schleife bis zur Abbruchsbedingung
    for (int i = 0; i < maxSchritte; i++){
      xWerte[i] = x0 + i*h;
      phiWerte[i] = status.phi;
      rWerte[i] = status.r;
      rkSchritt(status, parameter, h);

      // theta und thetadot Aufnahme nach jeder Umrundung
      if(status.phi >= 2*3.14159*k){
        k++;
        thetaWerte[i] = status.theta;
        thetadotWerte[i] = status.thetadot;
      }
    }

  // Rückgabe für eine grafische Wiedergabe
    return List::create(Named("x") = xWerte,Named("phi") = phiWerte,Named("r") = rWerte,
                        Named("theta") = thetaWerte,Named("thetadot") = thetadotWerte);
}
```

Exzentrizität $= 0$





Pho- bos

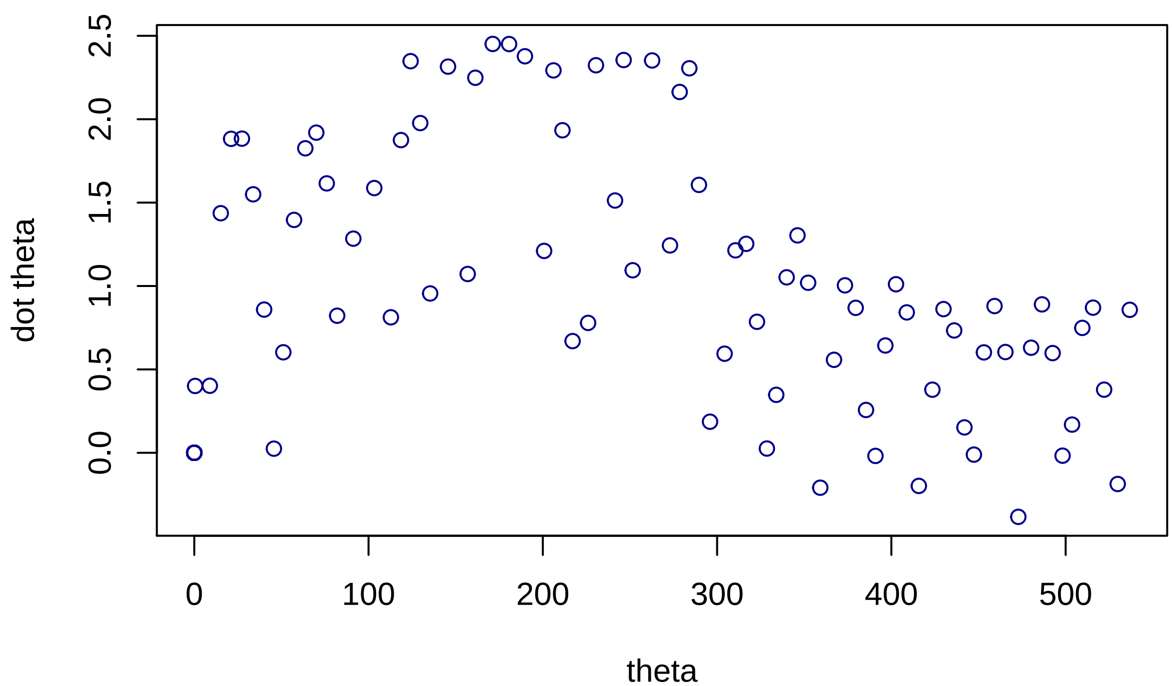


Abbildung 1: Entwicklungskurven für $R_0 = 2.9$

Fazit

Literatur