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:

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

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):

$$r(\phi) = \frac{p}{1 + \epsilon\cos\phi} = \frac{a(1-\epsilon^2)}{1 + \epsilon\cos\phi} \tag{3}$$

Und mit dem Drehimpuls und der Drehimpulserhaltung (QUELLE):

$$L = mr(\phi)^2 * \dot{\phi} = const \tag{4}$$

Somit können wir eine Bewegungsgleichung für ϕ finden und durch das Lösen sowohl die Zeitentwicklung für ϕ und r (in einheiten von a) finden.

$$\dot{\phi} = \frac{L}{m}\frac{(1 + 2\epsilon\cos\phi + \epsilon^2\cos\phi^2)}{(1-\epsilon^2)^2} \tag{5}$$

So wollen wir nun vorerst diese zweite Diffenzialgleichung lösen um damit die Erste zu lösen.

GRÖßEN: L; M; T

Runge-Kutta 2 Verfahren

Um die differenziellen Gleichungssysteme auszuwerten, verwenden wir das Runge-Kutta Verfahren nach

$$x_{i+1} = x_i + \frac{h}{2}[f(t_i, x_i) + f(t_i + h, x_i + hf(t_i, x_i))]. \tag{6}$$

Wobei sich unser f aus den vier Anteilen von S, I, R und V zusammensetzt. Wir werden die Auswirkungen der Schrittweite am Ende der Auswertung genauer betrachten.

Vorerst wählen wir die Schrittweite h = 1, da dies direkt den Tagen entspricht. Dabei bietet uns das Runge-Kutter Verfahren numerische Stabilität und weist mit einem Verfahrensfehler von $\mathcal{O}(h^2)$ einen kleineren Fehler als das Eulerverfahren auf.

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;
} 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.phi = parameter.LM*(1+2*parameter.epsilon*cos(alterStatus.phi)+parameter.epsilon*parame

}

// 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};

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

//[[Rcpp::export]]
Rcpp::List durchlauf(const int maxSchritte, const double h,
                           const double phi, const double r, const double theta,
                           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);
  // Quelltext
  Status status = {.phi = phi, .r = r, .theta = theta};
  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] = (1-parameter.epsilon*parameter.epsilon)/(1+status.phi*parameter.epsilon);
      rkSchritt(status, parameter, h);
    }

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

Test 1 für phi

Fazit

Literatur

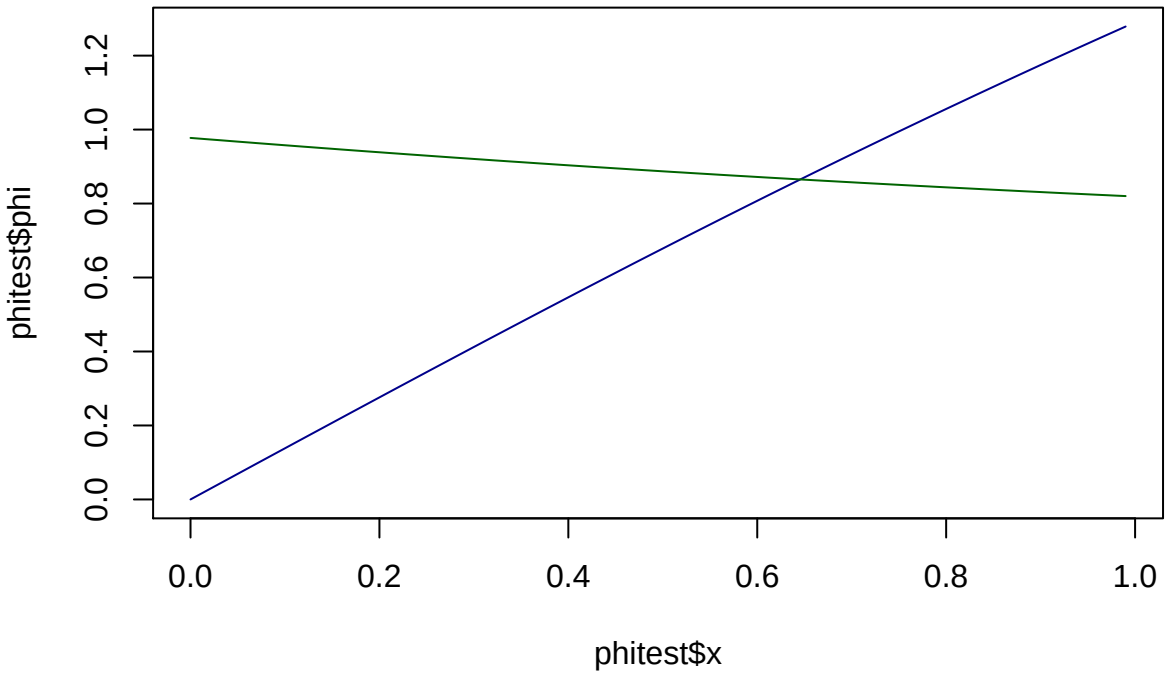

Abbildung 1: Entwicklungskurven für $R_0 = 2.9$