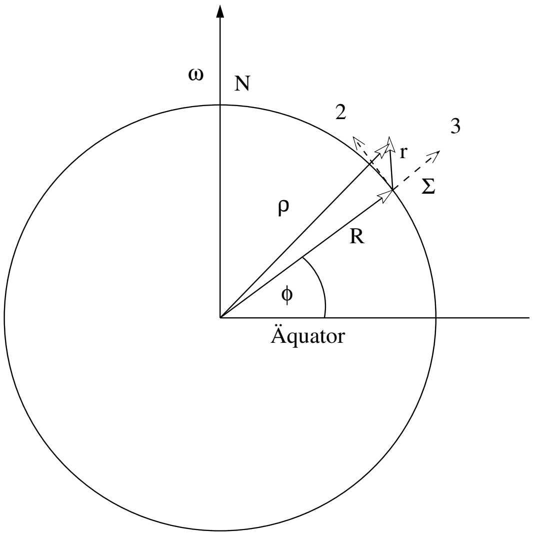
Übungsaufgaben **Theoretische Physik I**

WS 04/05 Prof. Dr. C. Gros

2. Übung Abgabe: Mi.,3.11.04
(bis 12:00, im Briefkasten)

Aufgabe 5



Auf der Erdoberfläche werde in einem Punkt der geographischen Breite ϕ ein kartesisches Koordinatensystem Σ angebracht mit den Achsen: 3 nach oben, 1 nach Osten und 2 nach Norden (siehe Abbildung). Die Winkelgeschwindigkeit der Erde ist $\omega = 7.27\ 10^{-5}\ s^{-1}$. $\underline{R}$ sei der Vektor vom Erdmittelpunkt zum Ursprung von Σ. $\circ$ bezeichne die Ableitung im beschleunigten System.

a.) Leiten Sie die Bewegungsgleichung

$$m\overset{\circ\circ}{\underline{r}} = m\,\underline{g} - m\,\underline{\omega} \times (\underline{\omega} \times \underline{R}) - 2m\,\underline{\omega} \times \overset{\circ}{\underline{r}} - m\,\underline{\omega} \times (\underline{\omega} \times \underline{r})$$

für eine Masse m am Ort $\underline{r}$ in Σ her.

b.) Für einen zur Zeit $t = 0$ in Σ ruhenden Massenpunkt erhält man durch Trägheitskräfte eine effektive Schwerebeschleunigung $\underline{\tilde{g}}$, die von $\underline{g}$ abweicht. Geben Sie diese Abweichung an.

c.) Wie hängt die auf m wirkende Corioliskraft von ϕ ab ?

d.) Korrigieren Sie Σ jetzt so, daß die 3-Achse senkrecht zur Meeresoberfläche steht. Wie lautet die Bewegungsgleichung für einen Punkt nahe der Erdoberfläche ?

Hinweis: Die Corioliskraft kann in guter Näherung aus c.) genommen werden. Vernachlässigen Sie Terme in ω^2 !

e.) Ein Massenpunkt werde aus der Höhe h fallengelassen. Lösen die Bewegungsgleichung unter der Annahme, daß sich die Ortskoordinaten in 1- und 2-Richtung nur wenig ändern. Bestimmen Sie die Fallzeit und die von der Erdrotation bewirkte Ostabweichung.

Hinweis: Es gilt $\overset{\bullet}{\underline{A}}=\overset{\circ}{\underline{A}}+\underline{\omega}\times\underline{A}$ für einen Vektor $\underline{A}$, wobei $\bullet$ die übliche Ableitung im Inertialsystem bezeichnet.

Aufgabe 6

Das Skalarprodukt zweier Vektoren $\vec{a}$ und $\vec{b}$ aus $\mathbb{R}^3$ ist durch $\vec{a}\cdot\vec{b}=\delta^{\alpha\beta}a^\alpha b^\beta = a^\alpha b^\beta$ gegeben, wobei wir die Konvention verwenden, dass über doppelt vorkommende Indizes zu summieren ist. Das Vektorprodukt zweier Vektoren $\vec{a}$ und $\vec{b}$ aus $\mathbb{R}^3$ ist durch $(\vec{a}\times\vec{b})^\alpha = \epsilon^{\alpha\beta\gamma}a^\beta b^\gamma$ gegeben. Hierbei ist

$$\epsilon^{\alpha\beta\gamma} := \begin{cases} 1 & , \ (\alpha,\beta,\gamma) \text{ gerade Permutation von } (1,2,3) \\ -1 & , \ (\alpha,\beta,\gamma) \text{ ungerade Permutation von } (1,2,3) \\ 0 & , \ \text{sonst} \end{cases}$$

Es gilt $\epsilon^{\alpha\beta\gamma}\epsilon^{\mu\nu\gamma}=\delta^{\alpha\mu}\delta^{\beta\nu}-\delta^{\alpha\nu}\delta^{\beta\mu}$.

a.) Beweisen Sie die “BAC-CAB” Relation: $\vec{a}\times(\vec{b}\times\vec{c})=\vec{b}\cdot(\vec{a}\cdot\vec{c})-\vec{c}\cdot(\vec{a}\cdot\vec{b})$.

b.) Zeigen Sie die Jacobi-Identität: $\vec{a}\times(\vec{b}\times\vec{c})+\vec{b}\times(\vec{c}\times\vec{a})+\vec{c}\times(\vec{a}\times\vec{b})=0$.

c.) Es gilt $|\vec{a}|=\sqrt{\vec{a}\cdot\vec{a}}$, $\vec{a}\cdot\vec{b}=|\vec{a}||\vec{b}|\cos\phi$ und $\vec{a}\times\vec{b}=|\vec{a}||\vec{b}|\sin\phi$, wobei ϕ der von $\vec{a}$ und $\vec{b}$ aufgespannte Winkel ist. Die geometrische Bedeutung von $\vec{a}\cdot(\vec{b}\times\vec{c})$ ist ein Volumen, das von den Vektoren $\vec{a}$, $\vec{b}$ und $\vec{c}$ beschrieben wird. Erläutern Sie dies näher.

Aufgabe 7

Die Programiersprachen C++ ist die Basis von Java und PHP. Sie ist eine moderne Programiersprache mit Klassen und Objekten. Die Beherschung dieser Sprache gehört zum Handwerkszeug des modernen Physikers.

Tippen Sie das umseitig folgende einfache und selbsterklärende C++ Programm ein und plotten Sie die Parabel.

```
/* October 2004                                        */
/* A simple program for computing and                  */
/* ploting parabolic trajectory                        */
/*                                                     */
/*   X(t)= X_0+V_x0*t                                  */
/*   Z(t)= Z_0+V_y0*t-g/m*t^2                          */
/*                                                     */

// includes

#include <iostream.h>   // (input/ouput)
#include <iomanip.h>    // (manipulating input/ouput with 'setw()' etc)
#include <math.h>       // (math's commands)
#include <fstream.h>    // (in/output on files)
#include <assert.h>     // ('assert()' calls abort if arg not true)

// Constant  definitions
// Numerical constants
const int Max_dimension=100;
const double  Delta_t=0.1;
// Dynamical constants
const double g=9.8;
const double m=1;
// Kinematical constants (initial values)
const double X_0=0.;
const double Z_0=0.;

// Type definitions
typedef double   VECTOR[Max_dimension];

main()
{
VECTOR  X,Z;             /*  coodinates of the point             */
VECTOR  time;            /*  time at points X Y                  */
double     V_x0,V_z0;    /*  initial velocities                  */
double     Flying_time;  /*  total time of flight T              */
double     Range;        /*  range of the X                      */
int        steps;        /*  number of points in the trajectory  */
 short int  test;        /*  test variable                       */

// input initial velocities;

cout << "type V_x0 and V_z0" <<endl;
cin >> V_x0 >> V_z0;
assert( (V_x0>0.) &&  (V_z0>0.) );
// computing trajectory;
```

```
steps=-1;

double t;
steps=-1;
test=0;
for (int i=0; i<Max_dimension && test==0  ;i++)
   {

   t=i*Delta_t;
   steps++;
   time[i]=t;
   X[i]=X_0+V_x0*t ;
   Z[i]=Z_0+V_z0*t-g/m*t*t;
   if (Z[i]<0.) test=1;
  }

steps--;
// computing Flying time and range
Flying_time=time[steps]    ;
Range      =X[steps]-X_0   ;

//  output on screen

cout<<"Flying time= "<< Flying_time<< endl;
cout<<"Range      = "<< Range      << endl;

// Output of trajectory in the file parabola.dat

ofstream  parabola("parabola.dat",ios::out);
parabola.precision(5);
parabola <<"# Parabolic trajectory  "
         <<"V_x0="<<V_x0<<"    "<<"V_y0="<<V_z0<<endl;

parabola <<"#      t"<<"      X(t)"<<"    Z(t)"<< endl;
for (int i=0; i<steps+1 ;i++)
 {parabola << setw(8) << time[i]
           << setw(8) << X[i]
           << setw(8) << Z[i] << endl;
 }
}  // end of main
```