Physik-Marathon 2024

– Aufgabe 15 –

(30. September – 13. Oktober)

Auf einem kugelrunden Ballon mit dem Radius $r_0 = 1\,\mathrm{m}$ leben zufrieden zwei kleine Tierchen. Eines Tages, als sich das Weibchen gerade am Nordpol der Kugel befindet und sich das Männchen gerade am Südpol aufhält, zur Zeit $t = 0$, beginnt ein böser Geist den Ballon aufzupusten, und zwar so, dass der Radius des Ballons um $u = \dot{r} = 1\,\mathrm{m\,s^{-1}}$ wächst. Im selben Moment beginnen beide Tierchen, entlang eines Meridians mit der Geschwindigkeit $v = 0{,}25\,\mathrm{m\,s^{-1}}$ aufeinanderzu zu laufen.

a) Wie lange dauert es, bis sich beide Tierchen auf dem Äquator der Kugel treffen? Berechne die Zeitdauer in Sekunden, gerundet auf ganze Sekunden!

b) Wie lange dauert es dagegen, wenn nur das Männchen mit derselben Geschwindigkeit v wie oben nordwärts läuft und das Weibchen am Nordpol verharrt? Berechne auch hier die Zeitdauer in Sekunden, gerundet auf ganze Sekunden!

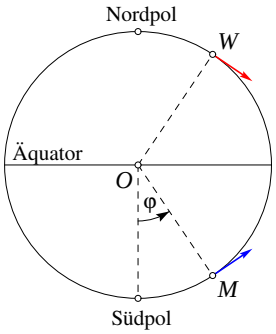
☞ Lösung und Punktverteilung auf der Rückseite.

Lösung von Aufgabe 15:

a) Wir betrachten das Geschehen in einem raumfesten Bezugssystem, in dem der Kugelmittelpunkt O ruht. Dann wächst der Radius der Kugel nach dem Gesetz

$$r(t) = r_0 + ut. \tag{1}$$

Die folgende Abbildung zeigt die Positionen und Geschwindigkeiten beider Tierchen zu einer Zeit $t > 0$:



Von O aus gesehen vollführt der Punkt auf der Kugeloberfläche mit dem gleichmäßig wachsenden Radius (1) eine Bewegung mit der Winkelgeschwindigkeit

$$\omega = \frac{v}{r} = \frac{v}{r_0 + ut} = \frac{\mathrm{d}\varphi}{\mathrm{d}t}. \tag{2}$$

Der Drehwinkel zählt entgegengesetzt zum Uhrzeigersinn, beginnend am Südpol (s. Abbildung oben). Seine Anfangsbedingung bei $t = 0$ lautet

$$\varphi(t = 0) = 0. \tag{3}$$

Nun stellt (2) eine Differenzialgleichung (DGL) 1. Ordnung für die Funktion $\varphi(t)$ dar:

$$\mathrm{d}\varphi = v\,\frac{\mathrm{d}t}{r_0 + ut}, \tag{4}$$

die leicht mit der Anfangsbedingung (3) integriert werden kann:

$$\int_0^{\frac{\pi}{2}} \mathrm{d}\varphi = \frac{\pi}{2} = v \int_0^{t_1} \frac{\mathrm{d}t}{r_0 + ut} = \frac{v}{u} \ln \frac{r_0 + ut_1}{r_0}. \tag{5}$$

Im linken Integral wurde als obere Grenze $\varphi = \frac{\pi}{2}$ gesetzt, da sich Männchen und Weibchen auf dem Äquator treffen. Gleichung (5) umgestellt nach t_1 ergibt mit den gegebenen Zahlenwerten

$$t_1 = \frac{r_0}{u}\left(\mathrm{e}^{\frac{\pi u}{2v}} - 1\right) = \mathbf{534\ s}. \tag{6}$$

b) Wenn das Männchen allein läuft, muss es den halben anstelle des Viertel-Vollwinkels laufen. Wir bekommen hier:

$$t_2 = \frac{r_0}{u}\left(\mathrm{e}^{\frac{\pi u}{v}} - 1\right) = \mathbf{286750\ s}. \tag{7}$$

Die benötigte Zeit bis zum Zusammentreffen ist jetzt also bedeutend größer als bei a).

Alternative Lösung:

Anstatt die bereits nach Variablen getrennte DGL analytisch zu integrieren (so wie es oben mit (5) getan wurde), können wir auch ein numerisches Verfahren hierauf anwenden. Das einfachste dieser Art ist das explizite EULER-Verfahren. Dazu werden in (4) die Zeiten und Winkel mit

$$t_i = i \cdot \Delta t, \quad \varphi_i = \varphi(t = t_i), \quad i = 0, 1, 2, \ldots \tag{8}$$

diskretisiert, wobei die Zeitschrittweite Δt fest gewählt wird. Die infinitesimalen Ausdrücke in (4) gehen dann mit der Vorwärtsdifferenz $\Delta\varphi = \varphi_{i+1} - \varphi_i$ über in die endlichen Ausdrücke

$$\Delta\varphi = \varphi_{i+1} - \varphi_i = \frac{v}{r_0 + u t_i}\,\Delta t. \tag{9}$$

Diese Gleichung nach φ_{i+1} umgestellt, ergibt mit der Anfangsbedingung $t_0 = 0$ und $\varphi_0 = 0$ bereits das Verfahren:

$$\varphi_{i+1} = \varphi_i + \frac{v}{r_0 + u(i\,\Delta t)}\,\Delta t, \quad i = 0, 1, 2, \ldots \tag{10}$$

$$\varphi_0 = 0, \quad t_0 = 0. \tag{11}$$

Die Rekursion (10) bricht ab, wenn der Winkel φ_{i+1}, etwa bei $i = n$, den Wert $\frac{\pi}{2}$ überschreitet. Dabei ist es kaum möglich, diesen Wert bei willkürlich gewähltem Δt genau zu treffen, sodass eine *lineare Interpolation* zwischen den letzten beiden Punkten (t_n, φ_n) und (t_{n+1}, φ_{n+1}) angebracht erscheint, um einen verbesserten Wert $\hat{t}$ für die Gesamtzeit zu erhalten:

$$\hat{t} = t_n + \left(\frac{\pi}{2} - \varphi_n\right) \frac{\Delta t}{\varphi_{n+1} - \varphi_n}. \tag{12}$$

Ein Programmfragment in `C`, welches das EULER-Verfahren (10) implementiert, sieht so aus:

```
double v = 0.25, r0 = 1.0, u = 1.0;
double phiold, phinew = 0.0, t, told, tnew = 0.0, dt = 0.00001;
unsigned long long i = 0ULL;
do {
  told = (double)i*dt;
  phiold = phinew;
  phinew = phiold + v/(r0+u*told)*dt;
  i++;
} while (phinew < 0.5*M_PI);
t = told + (_pi2-phiold)*dt/(phinew-phiold);
printf("dt = %10.6f  t = %.5f\n", dt, t);
```

Das Verfahren (10)–(12) liefert folgende Ergebnisse:

Δt [s]	$\hat{t}_1$ [s]	$\hat{t}_2$ [s] (Euler)	$\hat{t}_2$ [s] (impl. Tr.)
1,0	300,15695	160998,74419	265441,87846
0,1	508,00154	272538,35206	286511,69088
0,01	531,82144	285317,76235	286747,92358
0,001	534,22443	286606,94993	286750,28924
0,0001	534,46493	286735,97574	286750,31290
0,00001	534,48898	286748,87941	286750,31316
0,000001	534,49139	286750,17137	286750,31476
exakt	534,49166	286750,31314	286750,31314

Es ist zu erkennen, dass die Konvergenz bei $\hat{t}_1$ akzeptabel ist, bei $\hat{t}_2$ sind jedoch über $2{,}6 \cdot 10^{11}$ Schleifendurchläufe zu absolvieren, bis das Ergebnis in der geforderten Genauigkeit feststeht.

Hier lohnt es sich generell, nach genaueren Verfahren Ausschau zu halten. Eine Verbesserung bringt schon das simple *implizite Trapez-Verfahren*:

$$\varphi_{i+1} = \varphi_i + \frac{1}{2}\left(\frac{v}{r_0 + u(i\,\Delta t)} + \frac{v}{r_0 + u((i+1)\,\Delta t)}\right)\Delta t, \quad i = 0, 1, 2, \ldots \tag{13}$$

Der übliche Nachteil von impliziten Verfahren, dass in jedem Zeitschritt ggf. nichtlineare Gleichungen gelöst werden müssen, tritt hier nicht auf, da der Winkel φ in der rechten Seite von (4) und damit auch in (13) nicht vorkommt. Die Ergebnisse dieses Verfahrens sind in der vierten Spalte in obiger Tabelle aufgeführt. Der Aufwand beträgt nur rund ein Tausendstel gegenüber dem vorherigen Verfahren, das implizite Verfahren konvergiert also deutlich schneller.

Abschließend noch das Programmfragment in `C` zum Verfahren (13):

```
double v = 0.25, r0 = 1.0, u = 1.0;
double phiold, phinew = 0.0, t, told, tnew = 0.0, dt = 0.00001;
unsigned long long i = 0ULL;
do {
  told = (double)i * dt;
  phiold = phinew;
  tnew = told + dt;
  phinew = phiold + (v/(r0+u*tnew) + v/(r0+u*told)) * 0.5*dt;
  i++;
} while (phinew < 0.5*M_PI);
t = told + (_pi2 - phiold) * dt / (phinew - phiold);
printf("dt = %10.6f  t = %.5f\n", dt, t);
```

Dennoch muss betont werden, dass eine analytische Lösung, wie durch (6) gefunden, jeder numerischen Lösung hinsichtlich Genauigkeit und Geschwindigkeit, mit denen Ergebnisse errechnet werden, weit überlegen ist.

Punktverteilung:

- 0,7 Punkte, wenn nur eines der Ergebnisse (6) oder (7) richtig ist
- 1,0 Punkte, wenn beide Ergebnisse richtig sind