FKSZ14S2A4

Martin Murin, 4.IBDB
Gymnázium Jura Hronca

FKS A4 – Mravenisko

Obsah riešenia

Princíp

Na úvod si povedzme, čo sa tam vlastne deje. Keď mravec vybehne z mraveniska, zväčšuje svoj moment zotrvačnosti okolo osi otáčania kolotoča. Keďže jeho rýchlosť je iba v radiálnom smere, bude sa otáčať spolu s kolotočom, po ktorom beží, a bude mať teda rovnakú uhlovú rýchlosť ako tento kolotoč. Ako mravec beží ďalej a ďalej od stredu, zvyšuje svoj moment zotrvačnosti. V kombinácií s uhlovou rýchlosťou zvyšuje aj svoj moment hybnosti. Lenže celkový moment hybnosti v tejto sústave sa zachováva, lebo všetky sily na mravca pôsobia smerom do stredu a teda majú nulový moment. Z toho vyplýva, že si mravec nemôže moment hybnosti len tak vyrobiť, ale musí ho od niekade získať. Jediná vec, ktorá sa okrem mravca točí je kolotoč, takže moment hybnosti si „zoberie“ odtiaľ [1]. Toto spôsobí, že sa kolotoč začne spomaľovať.

Keď sa mravec dostane do určitej vzdialenosti, dostredivá sila bude väčšia ako trecia sila a v tom momente mravec odletí. Túto vzdialenosť budem ďalej označovať ako r_{max}. Keď mravec odletí, tak si vlastne zoberie svoj moment hybnosti zo sebou. Moment hybnosti sa premení na hybnosť, lebo už tam nie je žiadna dostredivá sila a ďalej nás nezaujíma. Moment

[1] Keď sa na mravca aj kolotoč pozrieme ako jednu sústavu, pohyb mravca od stredu osi otáčania zvyšuje celkový moment zotrvačnosti. Z toho, že moment hybnosti je súčin uhlovej rýchlosti a momentu zotrvačnosti vyplýva, že ak sa jedno väčší o nejaký faktor, druhé sa musí zmenšiť o ten istý faktor (aby sa zachoval celkový moment hybnosti).

Martin Murin, 4.IBDB
Gymnázium Jura Hronca

hybnosti mravca, ktorý odletel spočítam ako moment zotrvačnosti v bode, v ktorom odletel krát uhlová rýchlosť kolotoča v tom čase keď odletel – ona sa časom mení.

Maximálna povolená vzdialenosť:

$$F_{dostredivá} = F_{trecia}$$
$$m\omega^2 r = m\, g\, f$$
$$r_{max} = \frac{gf}{\omega^2}$$

Mravec je bod s hmotnosťou *m* a preto jeho moment zotrvačnosti okolo osi otáčania je:

$$I = m\, r^2$$
$$L = I\, \omega = m\, \omega\, r^2$$
$$L_{max} = \frac{mg^2 f^2}{\omega^3}$$

L_{max} je moment hybnosti, ktorý mravec odnesie, keď odletí. Čo sa teda deje je, že odchádzajúce mravce spomaľujú kolotoč a odletené mravce nemajú ďalej žiadny vplyv.

Keby som chcel vypočítať uhlovú rýchlosť, potreboval by som vedieť koľko mravcov už odletelo a koľko je na ceste a kde. K tomu by som však musel ešte vedieť, kedy a kde tie mravce odleteli, lebo moment hybnosti, ktorý zobrali so sebou závisí od miesta a času (uhlovej rýchlosti v tom čase) keď odleteli. Je to ťažké.

Komentár od [MB1]: Pri troche aproximácií, napr. že to všetko je spojité, to zase až také ťažké nie je ☺. Sú tam ešte aj ďalšie sily ako Coriolisova a Eulerova a patrilo by sa povedať, prečo sú malé resp. aký majú vplyv, pozri vo vzoráku komentár ku vzoráku ☺ Zato ten bod dole. Inak pekne.

Numerická simulácia

Rozhodol som sa naprogramovať jednoduchú simuláciu, ktorá mi dá aspoň predstavu, v akom čase by Enka mohla pozorovať pokles uhlovej rýchlosti na polovicu. Simuláciu som napísal v c++ a kompiloval v *C++ Builderi XE3* od *Embarcadero*.

Simulácia funguje nasledovne

- Zadefinujem počiatočné podmienky a spustím cyklus, ktorý beží až kým sa uhlová rýchlosť nezmenší na polovicu
- Zvýšim čas o jeden tik, čo je 0,01 sekundy, lebo toľko je interval medzi dvoma mravcami. Zároveň je to relatívne dobrá presnosť – skúšal som aj kratšie tiky, ale výsledok to skoro nezmenilo a trvalo to 10x tak dlho
- Vypočítam maximálny polomer r_{max} na základe uhlovej rýchlosti a konštánt
- Každého mravca posuniem o $v\, \Delta t$ ďalej od stredu

- Ak sa najvzdialenejší mravec dostal za r_{max}, jeho moment hybnosti si zapamätám a mravca zruším (používam dynamické pole)
- Cez sumu (for cyklus) vypočítam moment hybnosti mravcov, ktoré ešte neodleteli
- Z momentov hybností (mravcov na ceste a odletených mravcov) vypočítam novú uhlovú rýchlosť, ktorá je nižšia ako pôvodná. Koniec cyklu
- Výstup

Pre zaujímavosť (možno nutnosť) prikladám zdrojový kód ako obrázok. Text som priložil na koniec súboru.

FKSZ14S2A4.cpp

```
#pragma hdrstop
#pragma argsused

#include <tchar.h>
#include <stdio.h>
#include <iostream>
#include <vector>
#include <math.h>
using namespace std;

const double g = 9.81;       // [m*s^-2]
const double f = 0.5;        // friction coeficient
const double v = 0.08;       // [m/s]
const double m = 0.000005;   // [kg]
const double n = 100;        // [s^-1]
const double I = 5;          // moment of inertia  [kg*m^2]
const double dt = 0.01;      // tick [s]
const double L0 = 5;         // initial angular momentum [kg*m^2*s^-1]

class Ant {
    public:
        double r, t0;
        Ant(double);
};

Ant::Ant(double t_created){
    t0 = t_created;
    r = 0;
}

double newomega(double Lm){
    return (L0-Lm)/I;
}

int _tmain(int argc, _TCHAR* argv[])
```

```
int _tmain(int argc, _TCHAR* argv[])
{
    double omega = 1;       // angular velocity [rad/s]
    int flown = 0;
    double t = 0;
    double L_ants = 0;
    double L_away = 0;
    double r_max = g*f/(omega*omega);
    vector <Ant> runaway;
    while (omega > 0.5){
        t += dt;
        r_max = g*f/(omega*omega);

        // ants run
        for (unsigned int i = 0; i < runaway.size(); i++) {
            runaway.at(i).r += v*dt;
        }

        // another ant runs from the center
        Ant a(t);
        runaway.push_back(a);
        if (runaway.front().r >= r_max) {
            runaway.erase(runaway.begin());
            flown++;
            L_away += m*r_max*r_max*omega;
        }

        // calculate L_ants
        L_ants = 0;
        for (unsigned int i = 0; i < runaway.size(); i++) {
            L_ants += m*omega*pow(runaway.at(i).r, 2);
        }

        //calculate new omega
        omega = newomega(L_ants + L_away);
    }
    cout << "t = " << t << "\tomega = " << omega << "\n";
    cout << "max radius \t" << r_max << endl;
    cout << "flown = " << flown << endl;
    cout << "time of the last\t" << runaway.front().t0 << endl;
    system("PAUSE");
    return 0;
}
```

Martin Murin, 4.IBDB
Gymnázium Jura Hronca

Špeciálny prípad

Pred tým, než sa pustím do výstupu zo simulácie by som chcel ukázať, prečo jej verím. Ako som už písal, síce neviem analyticky spočítať celý príklad, ale viem spočítať prípad, keď mravce nebudú odletovať. V tomto prípade sa nemusím trápiť s tým, kedy koľko mravcov odletelo. Najprv si vypočítam moment zotrvačnosti všetkých mravcov spolu v čase *t*:

$$I_m(t) = m \sum r_i^2$$

$$I_m(t) = m \sum \left(v\,(t - t_i)\right)^2$$

Kde t_i je čas v ktorom bol vypustený i-ty mravec. Tento čas je rovný $\frac{i}{n}$, lebo *n* je počet mravcov za sekundu.

$$I_m(t) = m\,v^2 \sum \left(t - \frac{i}{n}\right)^2$$

$$I_m(t) = m\,v^2 \sum_{i=0}^{n\,t} \left(t^2 - 2\frac{t}{n}\,i + \frac{i^2}{n^2}\right)$$

Súčin *n t* je celkový počet mravcov. Dúfam, že ťa nepletiem, že vynechávam * ako operátor násobenia, ale takto to vyzerá lepšie + som zvyknutý na syntax z *Mathematica*, kde sa používa medzera. Poďme riešiť sumu – teraz neviem či tu mám detailne rozpisovať postup, alebo stačí výsledok, lebo toto je čistá matika (napíš mi prosím komentár, či toto treba robiť). Radšej to spravím, aby si nemal zbytočnú zámienku strhnúť mi body :D
Úprava výrazu sumy:

$$\sum_{i=0}^{n\,t} \left(t^2 - 2\frac{t}{n}\,i + \frac{i^2}{n^2}\right) = \sum_{t=0}^{n\,t} t^2 - \sum_{t=0}^{n\,t} 2\frac{t}{n}\,i + \sum_{t=0}^{n\,t} \frac{i^2}{n^2}$$

$$= t^2(n\,t+1) - 2\frac{t}{n}\frac{n\,t(n\,t+1)}{2} + \frac{1}{n^2}\frac{n\,t(n\,t+1)(2n\,t+1)}{6}$$

$$= \frac{t\,(n\,t+1)\,(2n\,t+1)}{6\,n}$$

Konečne som sa dopracoval k momentu zotrvačnosti mravcov v čase *t*.

$$I_m(t) = m\,v^2 \frac{t\,(n\,t+1)\,(2n\,t+1)}{6\,n}$$

Môžem začať riešiť špeciálny prípad. Celkový moment hybnosti sa zachováva, teda platí

Komentár od [MB2]: Ak by si povieš, že n je veľké, tak v tej sume prežije iba 1/3t^3, čo je to isté ako keby si integroval ☺

Martin Murin, 4.IBDB
Gymnázium Jura Hronca

$$L_{kolotoča} + L_{mravcov} =$$

Komentár od [MB3]: To môžeš využiť tak, že to zderivuješ potom je na pravej strane nula, dLkolotoc/dt si vypocitas aj dLmravce/dt odseparujes zintegrujes a mas vysledok. Vid vzorak. Numericke riesenie ale chvalim ☺

Kde c je nejaká konštanta, vlastne moment hybnosti kolotoča v čase *t = 0*. Túto rovnicu viem upraviť na nasledovné:

$$I_{kolotoča}\,\omega + I_m(t)\,\omega = c$$
$$\omega = \frac{c}{I_{kolotoča} + I_m(t)}$$

Otázka je, kedy sa uhlová rýchlosť zmenší na polovicu. Riešim teda rovnicu:

$$\omega(t) = \frac{1}{2}\omega(0)$$

V čase 0 žiadne mravce ešte nevyliezli a teda $I_m(0) = 0$

$$\frac{c}{I_{kolotoča} + I_m(t)} = \frac{\frac{1}{2}c}{I_{kolotoča}}$$
$$I_m(t) = I_{kolotoča}$$

Hľadám *t* pre ktoré toto platí. Na túto rovnicu som mohol prísť aj úvahou, že aby moment hybnosti zostal konštantný, pričom uhlová rýchlosť klesne na polovicu, moment zotrvačnosti sa musí zdvojnásobiť.

Nasledujúcu rovnicu treba vyriešiť pre *t*.

$$m\,v^2\,\frac{t\,(n\,t+1)\,(2n\,t+1)}{6\,n} - I_{kolotoča} = 0$$

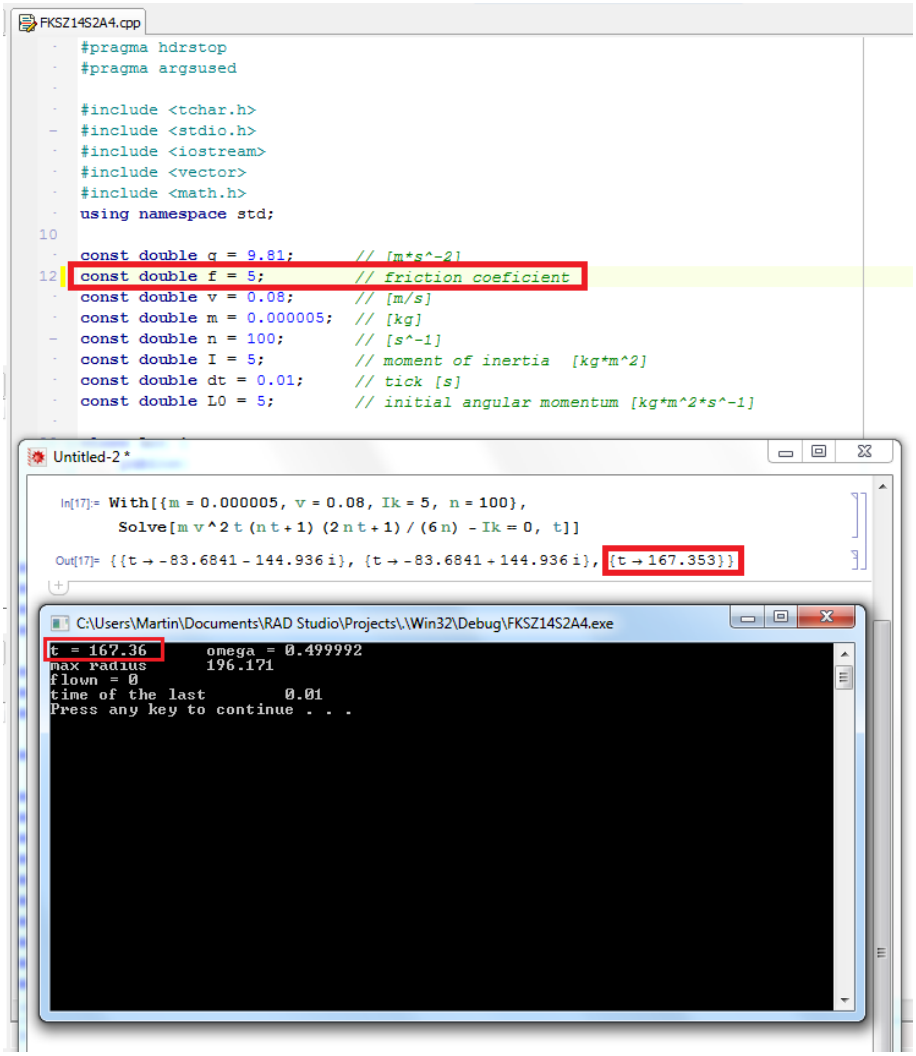
Táto rovnica je kubická pre *t* a preto ju neviem vyriešiť na papieri. Na jej vyriešenie som teda použil svetoznámy a úžasný *Wolfram Mathematica*. Za premenné *m*, *n* a *v* som teraz už dosadil čísla zo zadania, inak by bol výstup komplikovaný a plný odmocnín... verím, že aj kvôli tomuto ste do zadania pridali numerické hodnoty.

$$t = 167{,}353$$

Komentár od [MB4]: Tri platné číslice nie sú tri desatinné miesta za desatinou čiarkou, ak už pred nimi máš tri číslice;) Číže výsledok 167.

Keď som vo svojej simulácií nastavil koeficient trenia dosť vysoký na to, aby žiadne mravce nestihli odletieť, výstup zo simulácie bol presne tento istý, ako som práve vypočítal. To je dôvod, prečo si myslím, že moja simulácia funguje správne. Viď obrázok:

Martin Murin, 4.IBDB
Gymnázium Jura Hronca



Samozrejme, že moja simulácia má malé odchýlky kvôli tomu, že jeden tik nie je nekonečne malý. Ako vidíš, koeficient trenia som nastavil na *5*, očividne to stačí lebo na výstupe je premenná *flown = 0* – počet odletených mravcov.

Martin Murin, 4.IBDB
Gymnázium Jura Hronca

Všeobecný prípad

Skúšal som spočítať koľko mravcov v čase *t* odletelo a koľko ich je na ceste, ale nepodarilo sa mi k ničomu zaujímavému dostať. Preto sem dám výstupy z mojej simulácie, ktorá legitímnym spôsobom rieši problém pre konkrétne hodnoty.

Pri koeficiente trenia *f = 0.5* je čas *t = 175.81 s*.

Komentár od [MB5]: A nie 167?

FKSZ14S2A4.cpp

```
#pragma hdrstop
#pragma argsused

#include <tchar.h>
#include <stdio.h>
#include <iostream>
#include <vector>
#include <math.h>
using namespace std;

const double g = 9.81;       // [m*s^-2]
const double f = 0.5;        // friction coeficient
const double v = 0.08;       // [m/s]
const double m = 0.000005;   // [kg]
const double n = 100;        // [s^-1]
const double I = 5;          // moment of inertia  [kg*m^2]
const double dt = 0.01;      // tick [s]
const double L0 = 5;         // initial angular momentum [kg*m^2*s^-1]
```

C:\Users\Martin\Documents\RAD Studio\Projects\Win32\Debug\FKSZ14S2A4.exe

```
t = 175.81

omega = 0.499971
max radius    19.6188
flown = 1425
time of the last     14.26

Press any key to continue . . .
```

```
    double L_ants = 0;
    double L_away = 0;
    double r_max = g*f/(omega*omega);
    vector <Ant> runaway;
    while (omega > 0.5){
        t += dt;
        r_max = g*f/(omega*omega);
```

Martin Murin, 4.IBDB
Gymnázium Jura Hronca

Bonusová časť

Problém neriešim analyticky, takže ani tu nemám počítanie. Aspoň sa zamyslím nad tým, čo by sa malo stať. Znížením trenia mravce začnú skôr odlietať z kolotoča, r_{max} bude menšie ako v pôvodnom zadaní. Mravce budú so sebou „odnášať“ menej momentu hybnosti, lebo sa nedostanú tak ďaleko. Z toho vyplýva, že kolotoč sa bude točiť dlhšie, čo aj robí.

Pri koeficiente trenia *f = 0.05* je čas *t = 9748.84 s*.

Komentár od [MB6]: ok

FKSZ14S2A4.cpp

```
#pragma hdrstop
#pragma argsused

#include <tchar.h>
#include <stdio.h>
#include <iostream>
#include <vector>
#include <math.h>
using namespace std;

const double g = 9.81;      // [m*s^-2]
const double f = 0.05;          // friction coeficient
const double v = 0.08;      // [m/s]
const double m = 0.000005;  // [kg]
const double n = 100;       // [s^-1]
const double I = 5;         // moment of inertia  [kg*m^2]
const double dt = 0.01;     // tick [s]
const double L0 = 5;        // initial angular momentum [kg*m^2*s^-1]
```

```
C:\Users\Martin\Documents\RAD Studio\Projects\.\Win32\Debug\FKSZ14S2A4.exe
t = 9748.84
omega = 0.499999
max radius    1.96199
flown = 972431
time of the last     9724.32
Press any key to continue . . .
```

```
    double L_ants = 0;
    double L_away = 0;
    double r_max = g*f/(omega*omega);
    vector <Ant> runaway;
    while (omega > 0.5){
        t += dt;
        r_max = g*f/(omega*omega);
```

Martin Murin, 4.IBDB
Gymnázium Jura Hronca

Komentár k výstupu zo simulácie

Tu by som chcel vysvetliť, čo znamenajú výstupy, ktoré mali byť pôvodne iba informatívne pre mňa, ale už som ich tam nechal.

- *t* je celkový čas [v sekundách], kým kolotoč nespomalil svoju rotáciu na polovicu
- *omega* je uhlová rýchlosť [v radiánoch za sekundu] na konci, iba overujem či sa rovná polovici
- *max radius* je najväčšia vzdialenosť [v metroch] do ktorej sa mravce môžu dostať bez toho, aby odleteli – počítaná je v každom cykle odznovu a teda výstup je posledná hodnota
- *flown* – počet odletených mravcov
- *time of the last* – je čas, v ktorom bol vytvorený posledný mravec, ktorý odletel

Zdrojový kód

```
#pragma hdrstop
#pragma argsused

#include <tchar.h>
#include <stdio.h>
#include <iostream>
#include <vector>
#include <math.h>
using namespace std;

const double g = 9.81;       // [m*s^-2]
const double f = 0.5;        // friction coeficient
const double v = 0.08;       // [m/s]
const double m = 0.000005;   // [kg]
const double n = 100;        // [s^-1]
const double I = 5;               // moment of inertia  [kg*m^2]
const double dt = 0.01;           // tick [s]
const double L0 = 5;         // initial angular momentum [kg*m^2*s^-1]

class Ant {
      public:
            double r, t0;
            Ant(double);
};

Ant::Ant(double t_created){
      t0 = t_created;
      r = 0;
}

double newomega(double Lm){
      return (L0-Lm)/I;
}

int _tmain(int argc, _TCHAR* argv[])
{
```

```
	double omega = 1;	// angular velocity [rad/s]
	int flown = 0;
	double t = 0;
	double L_ants = 0;
	double L_away = 0;
	double r_max = g*f/(omega*omega);
	vector <Ant> runaway;
	while (omega > 0.5){
		t += dt;
		r_max = g*f/(omega*omega);

		// ants run
		for (unsigned int i = 0; i < runaway.size(); i++) {
			runaway.at(i).r += v*dt;
		}

		// another ant runs from the center
		Ant a(t);
		runaway.push_back(a);
		if (runaway.front().r >= r_max) {
			runaway.erase(runaway.begin());
			flown++;
			L_away += m*r_max*r_max*omega;
		}

		// calculate L_ants
		L_ants = 0;
		for (unsigned int i = 0; i < runaway.size(); i++) {
			L_ants += m*omega*pow(runaway.at(i).r, 2);
		}

		//calculate new omega
		omega = newomega(L_ants + L_away);
	}
	cout << "t = " << t << "\tomega = " << omega << "\n";
	cout << "max radius \t" << r_max << endl;
	cout << "flown = " << flown << endl;
	cout << "time of the last\t" << runaway.front().t0 << endl;
	system("PAUSE");
	return 0;
}
```