Московский авиационный институт

(национальный исследовательский университет)

**Лабораторные работа по курсу «Теоретическая механика»**

**“Определение реакций опор твердого тела и системы твердых тел”**

**“Анимация точки”**

**“Анимация системы”**

Выполнил студент группы № М7О-114БВ-24

Фельдман Лев Борисович

Преподаватель: Васькова В. С.

Оценки: \_\_\_\_\_\_

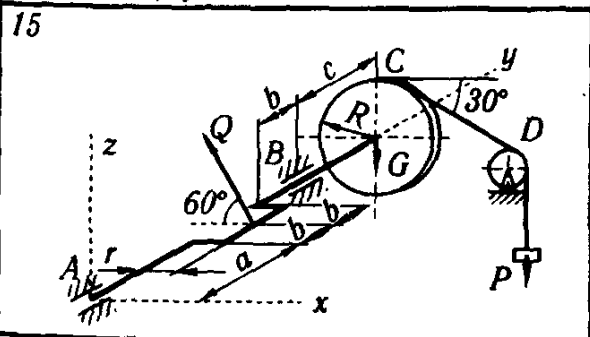
Дата: \_\_\_\_\_\_

Москва 2025

Лабораторная работа № 1. “Определение реакций опор твердого тела и системы твердых тел”

Вариант № 15

Задание: найти реакцию опор конструкции.



Код на MATLAB:

clc;

clear;

Q = [-cos(pi/3) 0 sin(pi/3)] \* 3 \* 10^3;

G = [0 0 1] \* 2 \* 10^3;

a = [0 60 0] \* 10^(-2);

b = [0 20 0] \* 10^(-2);

c = [0 40 0] \* 10^(-2);

R = 20 \* 10^(-2);

r = [5 \* 10^(-2) 0 0];

syms RAx RAz RBx RBz P;

PC = [cos(pi/6) 0 -sin(pi/6)] \* P;

RA=[RAx 0 RAz];

RB=[RBx 0 RBz];

rPC =[cos(pi/3) 0 sin(pi/3)] \* R;

V2A = [0 0 0];

V2Q = V2A + a + r + b;

V2B = V2A + a + 3\*b;

V2G = V2A + a + 3\*b + c;

V2PC = V2A + a + 3\*b + c + rPC;

mG=0:0.5:6;

mRAx=mG;

mRAz=mG;

mRBx=mG;

mRBz=mG;

mPz=mG;

for i=1:length(mG)

G=[0 0 -mG(i)]\*10^3;

% Составляем уравнения равновесия тела

eqnP=RA+RB+Q+G+PC==[0 0 0];

eqnM=cross(RA,V2A)+cross(RB,V2B)+cross(Q,V2Q)+cross(G,V2G)+cross(PC,V2PC)==[0 0 0];

%Решаем составленные уравнения

[FRAx, FRAz, FRBx, FRBz, FP]=solve([eqnP

eqnM],[RAx RAz RBx RBz P]);

mRAx(i)=eval(FRAx);

mRAz(i)=eval(FRAz);

mRBx(i)=eval(FRBx);

mRBz(i)=eval(FRBz);

mPz(i)=eval(FP);

end

figure

hold on

xlabel('G');

ylabel('mRAx,...')

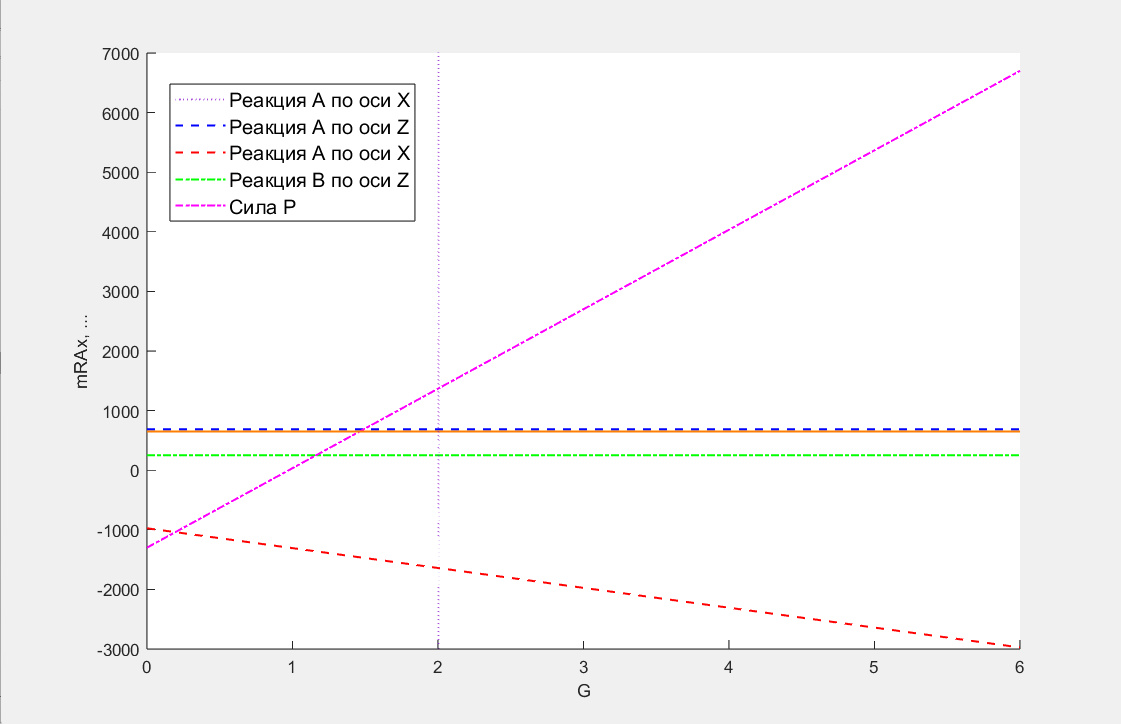
plot(mG,mRAx,'--r', 'LineWidth',2);

plot(mG,mRAz,'--b', 'LineWidth',2);

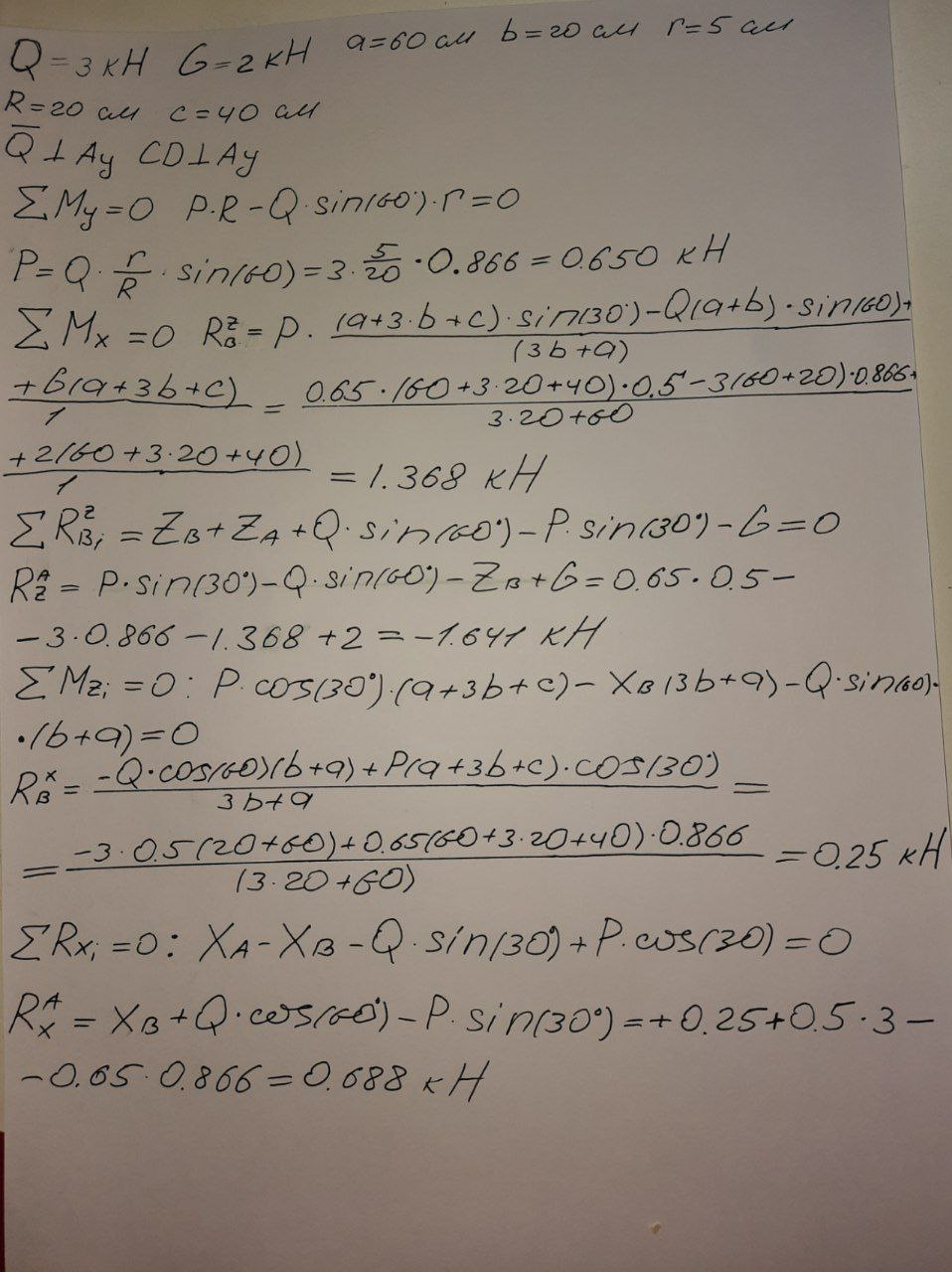
plot(mG,mRBx,':g', 'LineWidth', 1);

plot(mG,mRBz,':m', 'LineWidth', 1);

plot(mG,mPz, '-ok');



Расчеты вручную



Лабораторная работа №2 ‘Анимация точки’

Вариант № 14

r(t) = 2 +sin(12\*t) phi(t) = t +0.2\*cos(12\*t)

% 1. Параметры времени

Tmax = 10; % время моделирования, с

dt = 0.005; % шаг по времени, с

t = 0:dt:Tmax; % вектор времени

% 2. Задание полярных координат r(t) и phi(t)

r = 2 + sin(12\*t);

phi = t + 0.2\*cos(13\*t);

% 3. Перевод в декартовы координаты

x = r .\* cos(phi);

y = r .\* sin(phi);

% 4. Первая и вторая производные r и phi

dr = 12 \* cos(12\*t);

dphi = 1 - 0.2\*13 \* sin(13\*t);

d2r = -12^2 \* sin(12\*t);

d2phi = -0.2 \* 13^2 \* cos(13\*t);

% 5. Скорость и ускорение в декартовых проекциях

vx = dr .\* cos(phi) - r .\* sin(phi) .\* dphi;

vy = dr .\* sin(phi) + r .\* cos(phi) .\* dphi;

ax = d2r .\* cos(phi) ...

- 2\*dr .\* sin(phi) .\* dphi ...

- r .\* cos(phi) .\* dphi.^2 ...

- r .\* sin(phi) .\* d2phi;

ay = d2r .\* sin(phi) ...

+ 2\*dr .\* cos(phi) .\* dphi ...

- r .\* sin(phi) .\* dphi.^2 ...

+ r .\* cos(phi) .\* d2phi;

% 6. Расчёт центра кривизны

% скор² = vx^2 + vy^2; поперечный вектор v\_perp = [-vy, vx]

% скалярное произведение (v × a) = vx.\*ay - vy.\*ax

speed2 = vx.^2 + vy.^2;

crossVA = vx .\* ay - vy .\* ax;

cx = x - (vy .\* speed2) ./ crossVA;

cy = y + (vx .\* speed2) ./ crossVA;

% 7. Подготовка окна и фиксация области просмотра

figure;

hold on; grid on; axis equal;

plot(x, y, 'k:'); % вся траектория пунктиром

% вычисляем границы по r и добавляем небольшой запас

margin = 0.2;

Rmax = max(r) + margin;

axis([-Rmax, Rmax, -Rmax, Rmax]);

% 8. Инициализация графических объектов

hP = plot(0,0,'ro','MarkerSize',8,'MarkerFaceColor','r'); % точка

hV = quiver(0,0,0,0,'b','MaxHeadSize',2,'LineWidth',1.5); % скорость

hA = quiver(0,0,0,0,'r','MaxHeadSize',2,'LineWidth',1.5); % ускорение

hC = plot(0,0,'go','MarkerSize',8,'MarkerFaceColor','g'); % центр кривизны

legend('Траектория','Точка','Скорость','Ускорение','Центр кривизны', ...

'Location','best');

xlabel('x'); ylabel('y');

title('Анимация: точка, векторы скорости/ускорения и центр кривизны');

% 9. Масштабирование стрелок

scaleV = 0.05;

scaleA = 0.01;

% 10. Цикл анимации

for k = 1:length(t)

% обновляем положение точки

set(hP, 'XData', x(k), 'YData', y(k));

% обновляем вектор скорости

set(hV, 'XData', x(k), 'YData', y(k), ...

'UData', vx(k)\*scaleV, 'VData', vy(k)\*scaleV);

% обновляем вектор ускорения

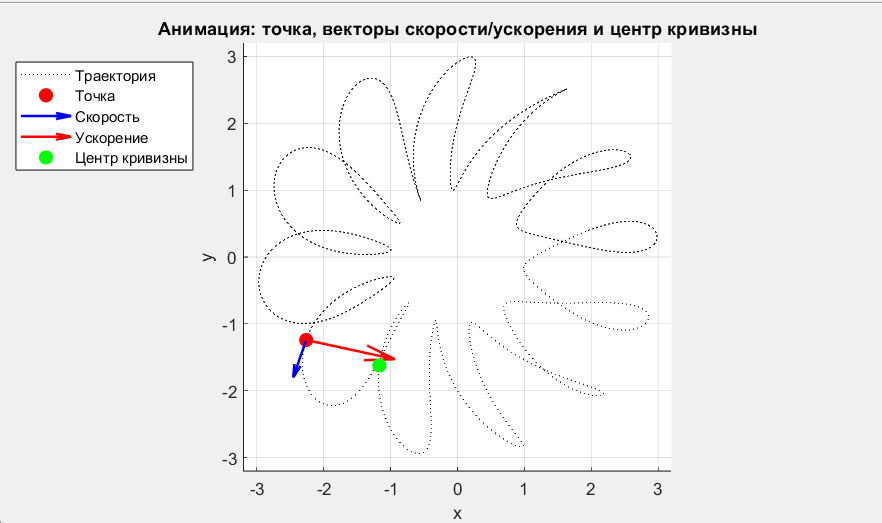
set(hA, 'XData', x(k), 'YData', y(k), ...

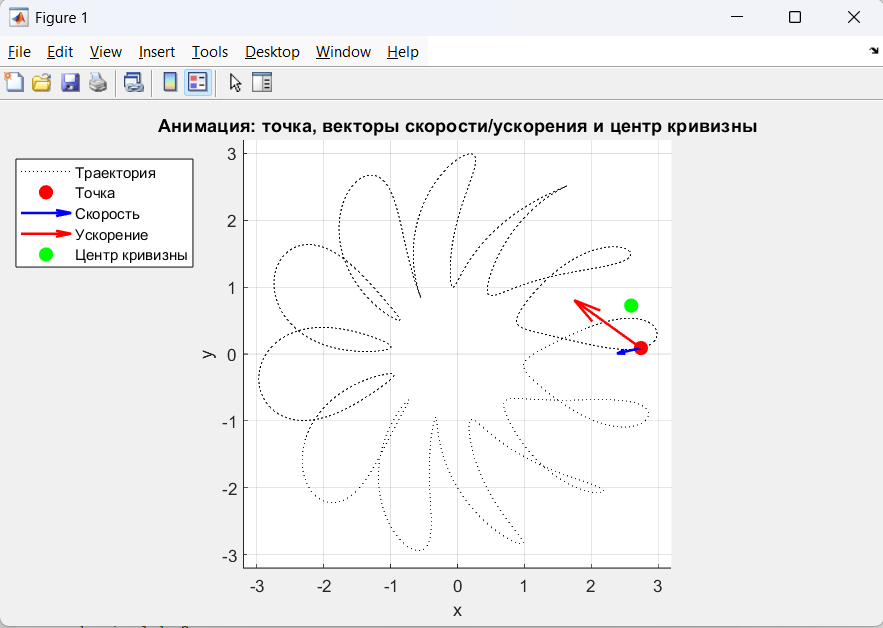
'UData', ax(k)\*scaleA, 'VData', ay(k)\*scaleA);

% обновляем центр кривизны

set(hC, 'XData', cx(k), 'YData', cy(k));

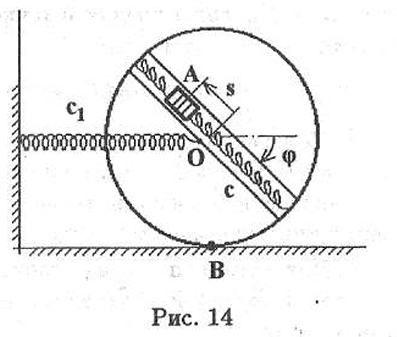
drawnow;





Лабораторная работа № 3. “Анимация системы”

Вариант № 14



% Анимация сферического тела на внешней пружине, внутреннего груза на пружинах

clear; clf; hold on; axis equal; grid on;

% Параметры системы

R = 1; % радиус шара

m\_b = 1; k\_ext = 10; % масса шара и жесткость внешней пружины

k\_int = 50; m\_i = 0.2; % жесткость и масса внутреннего груза

% Геометрия крепления

wall\_x = -5; % x-координата вертикальной балки (стены)

y\_beam = 0; % y-координата горизонтальной опоры (балки)

% Амплитуда и частота внешних колебаний

A\_ext = 1.5; omega\_ext = sqrt(k\_ext/m\_b); % амплитуда и угловая частота

% Амплитуда и частота внутренних колебаний

A\_int = 0.2; omega\_int = sqrt(k\_int/m\_i);

dt = 0.02; T = 10; t = 0:dt:T;

center0 = [wall\_x + 1 + A\_ext + R, R]; % стартовая точка при максимальном сжатии пружины

% --- Рисуем опоры ---

% Вертикальная балка (стена)

h\_wall = line([wall\_x, wall\_x], [-1, 4], 'LineWidth',6, 'Color',[0.4 0.4 0.4]);

% Горизонтальная балка (площадка)

beam\_x\_end = wall\_x + 1 + 2\*A\_ext + 2\*R + 2;

h\_beam = line([wall\_x-1, beam\_x\_end], [y\_beam, y\_beam], 'LineWidth',6, 'Color',[0.4 0.4 0.4]);

% Внешняя пружина (прикрепляется к шару)

h\_spring\_ext = line([wall\_x, center0(1)-R], [center0(2), center0(2)], 'LineWidth',2);

% Шар как круг через rectangle с Curvature=[1,1]

h\_ball = rectangle('Position',[center0(1)-R, center0(2)-R, 2\*R, 2\*R], ...

'Curvature',[1,1], 'EdgeColor','b','LineWidth',2);

% Внутренние пружины и груз внутри шара

h\_spring1 = line([center0(1), center0(1)], [center0(2)+0.9\*R, center0(2)+0.8\*R], 'LineWidth',1.5);

h\_spring2 = line([center0(1), center0(1)], [center0(2)-0.9\*R, center0(2)-0.8\*R], 'LineWidth',1.5);

h\_mass = rectangle('Position',[center0(1)-0.1, center0(2)-0.1, 0.2, 0.2], ...

'Curvature',0.1, 'FaceColor','r');

% Настройка осей

xlim([wall\_x-1, beam\_x\_end+1]); ylim([-1, 4]);

for i = 1:length(t)

% Полное гармоническое смещение без ограничений

s = A\_ext \* sin(omega\_ext \* t(i)); % смещение внешней пружины

theta = s / R; % угол прокрутки шара

center = [wall\_x + 1 + A\_ext - s + R, R];% центр шара движется от максимального сжатия

% Обновление внешней пружины, прикрепленной к шару

Ns = 16; x0 = wall\_x; x1 = center(1) - R; ys0 = center(2);

xs = linspace(x0, x1, Ns);

ys\_vec = ys0 + 0.15 \* sin(12 \* (xs - x0) / (x1 - x0) \* 2 \* pi);

set(h\_spring\_ext, 'XData', xs, 'YData', ys\_vec);

% Обновление позиции шара

set(h\_ball, 'Position', [center(1)-R, center(2)-R, 2\*R, 2\*R]);

% Внутренние колебания груза

y\_rel = A\_int \* sin(omega\_int \* t(i));

p\_mass = [0; y\_rel];

Rmat = [cos(theta), -sin(theta); sin(theta), cos(theta)];

p\_mass\_glob = Rmat \* p\_mass + center';

% Точки крепления внутренних пружин внутри шара

p1 = Rmat \* [0; R] + center';

p2 = Rmat \* [0; -R] + center';

% Обновление внутренних пружин и груза

set(h\_spring1, 'XData', [p\_mass\_glob(1), p1(1)], 'YData', [p\_mass\_glob(2), p1(2)]);

set(h\_spring2, 'XData', [p\_mass\_glob(1), p2(1)], 'YData', [p\_mass\_glob(2), p2(2)]);

set(h\_mass, 'Position', [p\_mass\_glob(1)-0.1, p\_mass\_glob(2)-0.1, 0.2, 0.2]);

drawnow;

pause(0.05);

end