МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В. Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа № 1**

по дисциплине: Системное моделирование

тема: «Поведение механических системы в статике»

Выполнил: ст. группы ПВ-223

Игнатьев Артур Олегович

Проверил:

доц. Полунин Александр Иванович

Белгород 2024г.

**Лабораторная работа №1**

**«Поведение механических системы в статике»**

**Цель работы:** Изучить особенности организации вычислительных процессов, связанные с погрешностями, приближенным характером вычислений на компьютерах современного типа, вычислительной устойчивостью.

**Вариант 3**

**Ход выполнения лабораторной работы:**

1. Разработать математическую модель, описывающую поведение элементов механической системы в статике.

Изображение выглядит как диаграмма, текст, линия, зарисовка

Автоматически созданное описание

Изображение выглядит как снимок экрана, линия, дизайн, диаграмма

Автоматически созданное описание

Изображение выглядит как линия, диаграмма, снимок экрана, Параллельный

Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, рукописный текст, типография

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

2. Разработать программу на основании математической модели и произвести расчёты.

clc; close all; clear;

fig = uifigure;

F = uislider(fig, "Position", [100 400 400 3], "ValueChangedFcn",@(F,event) Update(F.Value));

F.Limits = [0,10000];

F.Value = 0;

Update(F.Value);

GetBifurcationPoint();

T = uitable(fig, "Data", GetT);

function SetBP(newBP)

global BP

BP = newBP;

end

function x = GetBP

global BP

x = BP;

end

function SetT(newT)

global T

T = newT;

end

function x = GetT

global T

x = T;

end

function x = GetK1

x = 20000;

end

function x = GetK2

x = 40000;

end

function x = GetMass

x = 5;

end

function x = GetL

x = 1.2;

end

function x = GetN

x = 10;

end

function GetBifurcationPoint()

F = 0;

e = 0.00001;

step = 1000;

x0 = 0;

phi0 = 0;

[xr, phir, ex] = Newton(F, [x0 phi0]);

F\_i = [F];

X\_i = [xr];

Phi\_i = [phir];

i = 2;

while (step > e)

F = F + step;

[xr, phir, ex] = Newton(F, [x0 phi0]);

if (ex)

F\_i = [F\_i;F];

X\_i = [X\_i;xr];

Phi\_i = [Phi\_i;phir];

end

while (ex)

F = F + step;

[xr, phir, ex] = Newton(F, [x0 phi0]);

i = i + 1;

if (ex)

F\_i = [F\_i;F];

X\_i = [X\_i;xr];

Phi\_i = [Phi\_i;phir];

end

end

F = F - step;

step = step / 2;

end

SetT(table(F\_i, X\_i, Phi\_i));

SetBP([F xr phir]);

end

function Update(F)

syms x phi;

DrawPlots(F, [x phi]);

[xr, phir, ex] = Newton(F, [0 0]);

if (ex)

DrawPoint([xr phir], "g");

title("Static")

else

BP = GetBP;

str = "Bifurcation at F=" + string(BP(1)) + " x=" + string(BP(2)) + " phi=" + string(BP(3));

title(str, "Color","r");

end

hold off;

OutputInf(F, [xr, phir]);

end

function OutputInf(F, PointR)

Fr = GetFunctions(F, PointR(1), PointR(2));

str = "F=" + string(F) + " R=" + string(Fr(1)) + " M=" + string(Fr(2));

xlabel(str);

end

function DrawPlots(F, Args)

Fun = GetFunctions(F, Args(1), Args(2));

Draw(Fun(1), "r", [0 pi/2]);

Draw(Fun(2), "b", [0 pi/2]);

end

function Fun = GetFunctions(F, x, phi)

Fun = [GetR(F, x, phi); GetM(F, x, phi)];

end

function R = GetR(F, x, phi)

L=GetL;

m=GetMass;

k2=GetK2;

k1=GetK1;

n=GetN;

sin\_phi = sin(phi);

cos\_phi = cos(phi);

a = n + L \* sin\_phi - x;

b = L \* (1 - cos\_phi);

c = sqrt(a^2+b^2);

cos\_alpha = a / c;

dl1 = x;

dl2 = c - n;

Fy1 = k1 \* dl1;

Fy2 = k2 \* dl2;

R = -Fy1 + Fy2 \* cos\_alpha;

end

function M = GetM(F, x, phi)

L=GetL;

m=GetMass;

k2=GetK2;

k1=GetK1;

n=GetN;

sin\_phi = sin(phi);

cos\_phi = cos(phi);

a = n + L \* sin\_phi - x;

b = L \* (1 - cos\_phi);

c = sqrt(a^2+b^2);

cos\_alpha = a / c;

sin\_alpha = b / c;

cos\_phi\_minus\_alpha = cos\_alpha \* cos\_phi + sin\_alpha \* sin\_phi;

dl2 = c - n;

mg = m \* 9.81;

Fy2 = k2 \* dl2;

M1 = -mg \* sin\_phi \* L / 2;

M2 = F \* L;

M3 = -Fy2 \* cos\_phi\_minus\_alpha \* L;

M = M1 + M2 + M3;

end

function Draw(Fun, Color, Range)

graph(1) = ezplot(Fun, Range); hold on; grid on;

set(graph(1), "Color", Color);

end

function DrawPoint(Point, Color)

str = "(" + string(Point(1)) + ";" + string(Point(2)) + ")";

text(Point(2), Point(1), str);

plot(Point(2), Point(1),"\*", "Color", Color);

end

function Der = GetDeriviativeRx(F, Point)

e = 0.000000001;

Der = (GetR(F, Point(1) + e, Point(2)) - GetR(F, Point(1), Point(2))) / e;

end

function Der = GetDeriviativeRphi(F, Point)

e = 0.000000001;

Der = (GetR(F, Point(1), Point(2) + e) - GetR(F, Point(1), Point(2))) / e;

end

function Der = GetDeriviativeMx(F, Point)

e = 0.000000001;

Der = (GetM(F, Point(1) + e, Point(2)) - GetM(F, Point(1), Point(2))) / e;

end

function Der = GetDeriviativeMphi(F, Point)

e = 0.000000001;

Der = (GetM(F, Point(1), Point(2) + e) - GetM(F, Point(1), Point(2))) / e;

end

function w = GetW(F, Point0)

dRx = GetDeriviativeRx(F, Point0);

dRphi = GetDeriviativeRphi(F, Point0);

dMx = GetDeriviativeMx(F, Point0);

dMphi = GetDeriviativeMphi(F, Point0);

w = [dRx dRphi; dMx dMphi];

end

function LessEps = IsLessEps(A, Eps)

LessEps = abs(A) <= Eps;

end

function GreaterEps = IsGreaterEps(A, Eps)

GreaterEps = abs(A) > Eps;

end

function Delta = GetDelta(F, Point0)

Delta = linsolve(GetW(F, Point0),GetFunctions(F, Point0(1), Point0(2)));

end

function [xr, phir, ex] = Newton(F, Point0)

e = 0.00000001;

Point(1) = Point0(1);

Point(2) = Point0(2);

k = 1;

Delta = GetDelta(F, Point);

while and(and(and(not(and(IsLessEps(GetM(F, Point(1), Point(2)), e), IsLessEps(GetR(F, Point(1), Point(2)), e))), Point(1) >= 0), Point(2) >= 0), Point(2) <= pi/2)

Point = Point - Delta;

Delta = GetDelta(F, Point);

k = k + 1;

if (k > 1000)

break;

end

end

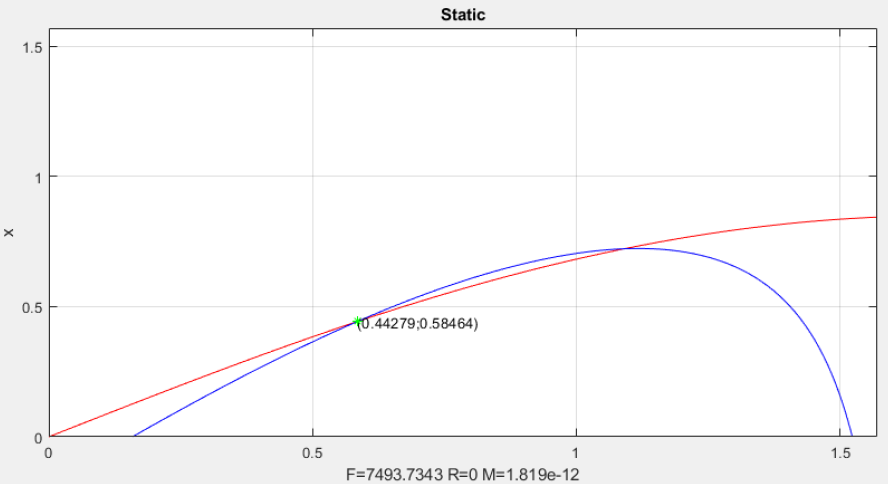
xr = Point(1);

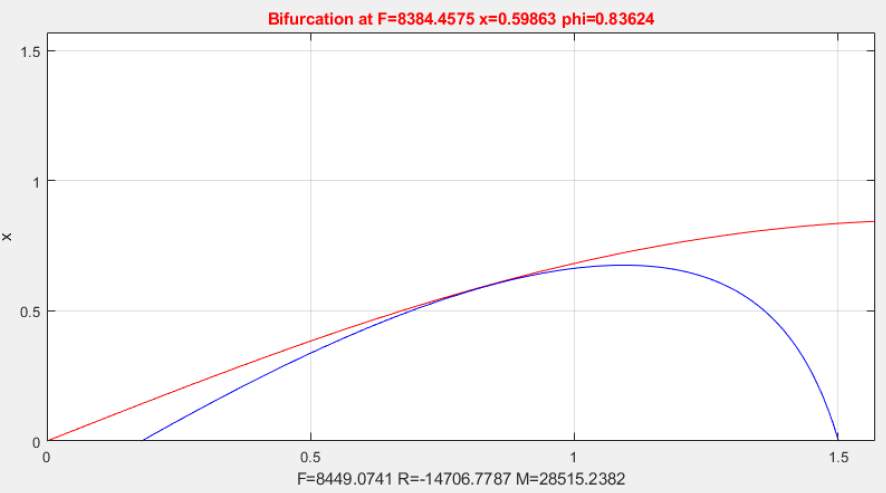
phir = Point(2);

ex = and(IsLessEps(GetM(F, Point(1), Point(2)), e), IsLessEps(GetR(F, Point(1), Point(2)), e));

end

Работа программы:





Изображение выглядит как текст, снимок экрана, число

Автоматически созданное описание

**Вывод:** в ходе выполнения лабораторной работы были изучены особенности организации вычислительных процессов, связанные с погрешностями, приближенным характером вычислений на компьютерах современного типа, вычислительной устойчивостью.