НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп’ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

**Лабораторна робота № 2.4**

**з навчальної дисципліни**

**"Математичні моделі динамічних систем"**

Тема: ДОСЛІДЖЕННЯ ДИНАМІКИ ПОВЗДОВЖНЬОГО РУХУ ЛІТАКА В ТУРБУЛЕНТНІЙ АТМОСФЕРІ

Виконавець: студент групи УС-213 Дзиговський В.І.

Київ 2020

**ДОСЛІДЖЕННЯ ДИНАМІКИ ПОВЗДОВЖНЬОГО РУХУ ЛІТАКА В ТУРБУЛЕНТНІЙ АТМОСФЕРІ**

**Мета лабораторної роботи**

1. Вивчити фізичну суть і характеристики атмосферної турбулентності та її вплив на динаміку польоту літака.

2. Вивчити особливості детермінованого та стохастичного моделювання атмосферної турбулентності.

3. Дослідити вплив атмосферної турбулентності різної інтенсивності на якість автоматичної стабілізації заданої висоти польоту літака.

## **Стислі теоретичні відомості**

Рух літака в повітряному просторі відбувається під дією тяги двигунів, сили тяжіння та аеродинамічних сил, зумовлених взаємодією літака з повітряним середовищем. На тягу двигуна та аеродинамічні сили значно впливають різноманітні метеорологічні чинники: вітер, температура, тиск, густина та вологість повітря, причому, вітер – більшою мірою.

Вітром у широкому значенні називається будь-який рух повітря відносно земної поверхні. Він може містити як великомасштабні горизонтальні та постійні за швидкістю переміщення повітряних мас, так і маломасштабні, невпорядковані переміщення – турбулентність. І якщо перші значно впливають, переважно, на вирішення навігаційних завдань, то політ у турбулентній атмосфері пов’язаний з появою перевантажень, лінійних коливань центра мас літака та кутових коливань відносно центра мас («бовтанки»).

Отже, атмосферна турбулентність (турбулентний вітер) – це хаотичні, невпорядковані пориви вітру, які мають просторову структуру і високу частоту зміни амплітуди і напрямку.

Причинами атмосферної турбулентності є взаємодія (тертя) висхідних і низхідних потоків повітря, що викликає утворення повітряних вихорів, взаємодія (тертя) повітряного потоку і об’єктів на землі, що також викликає утворення вихорів, процеси хмароутворення тощо.

Атмосферна турбулентність має шарувату структуру і виникає в атмосфері у вигляді шарів товщиною до одного кілометра і довжиною до декількох десятків кілометрів, переважно, на невеликих висотах.

Вектор швидкості турбулентного вітру має три складових:



де: – повздовжня складова; – вертикальна складова; – бічна складова.



Повздовжня складова турбулентного вітру викликає зміну повітряної швидкості літака, що призводить до зміни величини швидкісного напору і до зміни всіх сил і моментів, що діють на літак у польоті. Проте через велику інерційність літака по швидкості вплив її зміни, викликаної дією повздовжньої складової турбулентного вітру, на дискомфорт пасажирів та навантаження на конструкцію літака, є незначними.



Більш суттєвим є вплив вертикальної та бічної складових.

Вертикальні пориви турбулентного вітру зі швидкістю викликають зміни кута атаки (додатні або від’ємні за знаком):



0 ,



де: *V0* – швидкість польоту літака до дії поривів вітру.

При зміні кута атаки на величину змінюються коефіцієнти сили лобового опору , піднімальної сили та повздовжнього моменту , що призводить до зміни самих сил та моменту:



0



0



0



При цьому найбільш важливим є вплив вертикальних поривів турбулентного вітру на величину піднімальної сили, тому що вони викликають зміні вертикального перевантаження:

0,



викликаючи дискомфорт для пасажирів та навантаження на конструкцію літака та систему управління.

Ці явища підсилюються розкачуванням літака по тангажу під дією повздовжнього моменту 0.



Бічні пориви турбулентного вітру зі швидкістю викликають зміни кута ковзання (додатні або від’ємні за знаком):



,



де: *V0* – швидкість польоту літака до дії поривів вітру.

При зміні кута ковзання на величину змінюються коефіцієнти бічної сили, поперечного та шляхового моментів, що призводить до зміни самих сили та моментів:



0



0



0



Під дією поперечного та шляхового моментів, викликаних бічними поривами турбулентного вітру, літак розкачується по крену та рисканню, викликаючи дискомфорт пасажирів та навантаження на конструкцію та систему управління.

Явище дискомфорту поглиблюється дією бічного перевантаження:

0



Викликана дією вертикальної складової турбулентного вітру «бовтанка» вважається слабкою (не більше одного бала за інтенсивністю), якщо зміни вертикального перевантаження під час польоту в турбулентній атмосфері *Δny* ≤ ±0,2 , помірною (два бали), якщо ±0,2 < *Δny* ≤ ±0,5 , сильною (три бали), якщо ±0,5 < *Δny* ≤ ±1,0 , та штормовою (чотири бали), якщо *Δny* > ±1,0 .

Шляхом численних льотних експериментів було встановлено, що пілоти вважають інтенсивність турбулентності слабкою, якщо в результаті впливу її на літак середньоквадратичне значення зміни вертикального перевантаження   
 ≤ 0,1 , помірною, якщо 0,1 < ≤ 0,2 , сильною, якщо 0,2 < ≤ 0,3 , та дуже сильною, якщо > 0,3 .



При моделюванні атмосферної турбулентності застосовуються детермінований метод, при якому турбулентність представляється у вигляді послідовності дискретних різнополярних імпульсів фіксованої форми, та стохастичний метод, при якому турбулентність представляється у вигляді стохастичного (ймовірнісного) процесу.

Стохастичний метод, який дозволяє моделювати атмосферну турбулентність значно точніше, ніж метод дискретних імпульсів, побудований на основі положень теорії ймовірності та математичної статистики. При його застосуванні робляться такі припущення:

1) Поле швидкостей турбулентного вітру вважається «замороженим», тобто його ймовірнісні характеристики не змінюються під час польоту.

2) Поле швидкостей турбулентного вітру вважається однорідним та ізотропним, тобто його ймовірнісні характеристики однакові для всіх його складових та не залежать від напрямку руху літака.

При розробці стохастичної математичної моделі атмосферної турбулентності використовуються апарат кореляційних функцій або спектральних щільностей, між якими існує однозначна відповідність на основі перетворення Фурьє.

Стохастичні математичні моделі складових атмосферної турбулентності отримують за допомоги так званих формуючих фільтрів, на вхід яких подаються сигнали генератора (імітатора) «білого шуму».

«Білий шум» це узагальнений випадковий процес з рівномірно розподіленим за всіма частотами спектром, при їх зміні від 0 до ∞, повною відсутністю кореляції між амплітудами і частотами, нульовим математичним очікуванням і спектральною щільністю:



Статична обробка результатів досліджень здійснюється на основі відомих співвідношень теорії ймовірності та математичної статистики.

Результати досліджень показують, що в більшості випадків розподіл миттєвих значень швидкості вітру і параметрів руху літака в турбулентній атмосфері з достатньою для практики точністю підпорядковується нормальному закону (закону Гауса).

**Математична модель вертикальної складової турбулентного вітру**

Результати досліджень атмосферної турбулентності показують, що вертикальна складова турбулентного вітру має такий вираз спектральної щільності :



де: – повітряна швидкість польоту; = 300 м – масштаб вертикальної складової турбулентного вітру – параметр, який характеризує максимальний лінійний розмір турбулентної зони, в межах якого відчувається взаємозв’язок між швидкостями вітру в різних точках; – середньоквадратичне значення швидкості вертикальної складової турбулентного вітру.



На підставі наведеного вище виразу спектральної щільності можна отримати математичну модель вертикальної складової турбулентного вітру у вигляді системи таких диференціальних рівнянь:



де: – вертикальна складова швидкості турбулентного вітру; – проміжна змінна; – середньоквадратичне значення швидкості вертикальної складової турбулентного вітру; – сигнали генератора «білого шуму»; – крок чисельного інтегрування диференціальних рівнянь.



**Закон управління САУ в режимі автоматичної стабілізації заданої висоти**

Спрощений закон управління системи автоматичного управління (САУ) в режимі автоматичної стабілізації заданої висоти має такий вигляд:



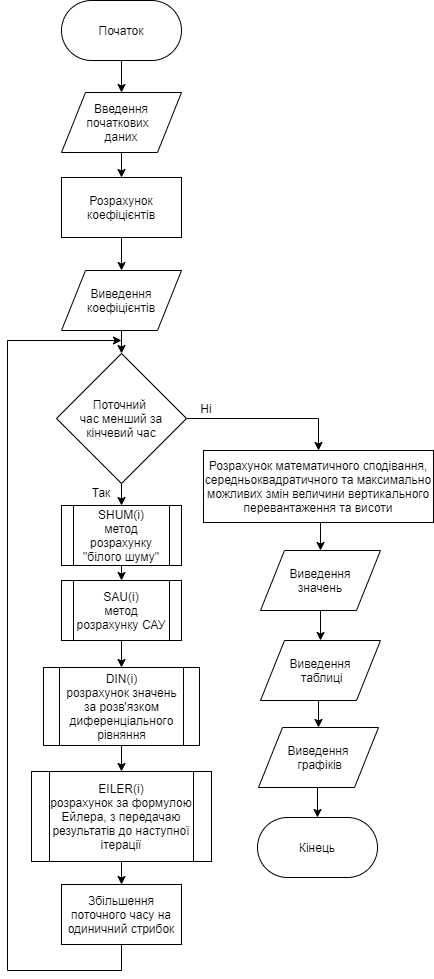
100 0,1 град/м



0,4 град с/м 3,0 с



**Алгоритм та програмна реалізація на Java**



import java.awt.\*;

import java.awt.event.\*;

import java.util.ArrayList;

import javax.swing.\*;

public class Lab extends JFrame{

public double S = 201.45, ba = 5.285, G = 73000, xt = 0.24, Iz = 660000, V = 190, H = 6400, p = 0.0636, a = 314.34, g = 9.81,

Cy = -0.28, Cya = 5.9, Cyd = 0.2865, Cx = 0.033, mz0 = 0.22, mzwz = -13.4, mza\_ = -4, mza = -1.95, mzd = -0.92;

public double C1, C2, C3, C4, C5, C6, C9, C16, abal, Cybal, dbal, mx, Sigma, xmax;

public TextField TC1, TC2, TC3, TC4, TC5, TC6, TC9, TC16, Tabal, TCybal, Tdbal, Tmx1, TSigma1, Txmax1 , Tmx2, TSigma2, Txmax2;

public double T = 0, DT = 0.01, TD = 0, TF = 300, DD = 5;

public Button TT, TSig, TDV, Ta, Tteta, TH, TNY;

public double MW1 = 0, MW2 = 0, MS1 = 0, MS2 = 0, MAX1 = 0, MAX2 = 0;

public double AV, hi = 0, Kh = 0.1, Kh\_D = 0.4, kwz = 3, Hz, Sigma\_Wy, Ry = 0.324;

TextField Table[][];

double[] min = {0,0};

double[] max = {0,0};

Scrollbar SB;

Graphics graph;

public class Data{

public double[] Y = new double[7];

public double[] X = new double[7];

public double DV, NY;

public Data(double Y1, double Y2, double Y3, double Y4, double Y5, double Y6, double Y7) {

Y[0] = Y1;

Y[1] = Y2;

Y[2] = Y3;

Y[3] = Y4;

Y[4] = Y5;

Y[5] = Y6;

Y[6] = Y7;

}

}

public ArrayList<Data> Database = new ArrayList();

public Lab(String s){

super(s);

setLayout(null);

setSize(1545, 830);

setVisible(true);

this.setLocation(0, 0);

this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

this.setResizable(false);

graph = getGraphics();

TC1 = CreatCTF("", 60, 600);

TC2 = CreatCTF("", 240, 600);

TC3 = CreatCTF("", 420, 600);

TC4 = CreatCTF("", 600, 600);

TC5 = CreatCTF("", 60, 650);

TC6 = CreatCTF("", 240, 650);

TC9 = CreatCTF("", 420, 650);

TC16 = CreatCTF("", 600, 650);

Tabal = CreatCTF("", 60, 700);

TCybal = CreatCTF("", 240, 700);

Tdbal = CreatCTF("", 420, 700);

Tmx1 = CreatCTF("", 600, 700);

TSigma1 = CreatCTF("", 40, 750);

Txmax1 = CreatCTF("", 190, 750);

Tmx2 = CreatCTF("", 340, 750);

TSigma2 = CreatCTF("", 490, 750);

Txmax2 = CreatCTF("", 640, 750);

TT = CreateBT("T", 22, 50, 80, 40);

TSig = CreateBT("σ", 102, 50, 80, 40);

TDV = CreateBT("δв", 182, 50, 80, 40);

Ta = CreateBT("α", 262, 50, 110, 40);

Tteta = CreateBT("θ", 372, 50, 110, 40);

TH = CreateBT("H", 482, 50, 110, 40);

TNY = CreateBT("ny", 592, 50, 93, 40);

Table = new TextField[16][7];

for(int i = 0; i < 16; i++) {

for(int j = 0; j < 7; j++) {

Table[i][j] = new TextField("");

Table[i][j].setFont(new Font("TimesRoman", Font.PLAIN, 16));

Table[i][j].setEditable(false);

add(Table[i][j]);

}

Table[i][0].setBounds(22, 90+i\*25, 80, 25);

Table[i][1].setBounds(102, 90+i\*25, 80, 25);

Table[i][2].setBounds(182, 90+i\*25, 80, 25);

Table[i][3].setBounds(262, 90+i\*25, 110, 25);

Table[i][4].setBounds(372, 90+i\*25, 110, 25);

Table[i][5].setBounds(482, 90+i\*25, 110, 25);

Table[i][6].setBounds(592, 90+i\*25, 93, 25);

}

SB = new Scrollbar(Scrollbar.VERTICAL);

SB.setBounds(685, 50, 20, 440);

SB.setBackground(Color.GRAY);

SB.setMinimum(1);

add(SB);

double m = G/g;

Cybal = 2\*G/(S\*p\*Math.pow(V, 2));

TCybal.setText(Double.toString(Cybal));

abal = 57.3\*((Cybal - Cy)/Cya);

Tabal.setText(Double.toString(abal));

dbal = -57.3\*((mz0 + (mza\*abal/57.3) + Cybal\*(xt - 0.24))/mzd);

Tdbal.setText(Double.toString(dbal));

C1 = -(mzwz/Iz)\*S\*ba\*ba\*p\*(V/2);

TC1.setText(Double.toString(C1));

C2 = -(mza/Iz)\*S\*ba\*p\*V\*V/2;

TC2.setText(Double.toString(C2));

C3 = -mzd/Iz\*S\*ba\*p\*V\*V/2;

TC3.setText(Double.toString(C3));

C4 = (Cya+Cx)/m\*S\*p\*V/2;

TC4.setText(Double.toString(C4));

C5 = -mza\_/Iz\*S\*ba\*ba\*p\*V/2;

TC5.setText(Double.toString(C5));

C6 = V/57.3;

TC6.setText(Double.toString(C6));

C9 = Cyd/m\*S\*p\*V/2;

TC9.setText(Double.toString(C9));

C16 = V/57.3/g;

TC16.setText(Double.toString(C16));

Database.add(new Data(0, 0, 0, 0, 0, 0, 0));

Hz=20.f; Sigma\_Wy=0.0f;

//Hz=0.f; Sigma\_Wy=0.5f;

//Hz=0.f; Sigma\_Wy=1.0f;

//Hz=0.f; Sigma\_Wy=3.0f;

while(T<=TF+DT) {

int i = (int)(Math.round(T/DT));

SHUM(i);

SAU(i);

DIN(i);

EILER(i);

MW1 += Database.get(i).NY;

MW2 += Database.get(i).Y[4];

T += DT;

CheckMaxMin(i);

}

MW1/=T/DT;

MW2/=T/DT;

for (int i=0;i<T/DT;i++) {

MS1 += Math.pow((Database.get(i).NY-MW1), 2);

MS2 += Math.pow((Database.get(i).Y[4]-MW2), 2);

}

MS1=Math.sqrt(1/(T/DT-1)\*MS1);

MS2=Math.sqrt(1/(T/DT-1)\*MS2);

MAX1=Math.abs(MW1)+2\*MS1; MAX2=Math.abs(MW2)+2\*MS2;

Tmx1.setText(Double.toString(MW1));

Tmx2.setText(Double.toString(MW2));

TSigma1.setText(Double.toString(MS1));

TSigma2.setText(Double.toString(MS2));

Txmax1.setText(Double.toString(MAX1));

Txmax2.setText(Double.toString(MAX2));

for(int i = 0; i < 16; i++) {

Table[i][0].setText(Double.toString((i+1)\*DD));

Table[i][1].setText(Double.toString(Database.get((i+1)\*(int)(DD/DT)).Y[5]));

Table[i][2].setText(Double.toString(Database.get((i+1)\*(int)(DD/DT)).DV));

Table[i][3].setText(Double.toString(Database.get((i+1)\*(int)(DD/DT)).Y[3]));

Table[i][4].setText(Double.toString(Database.get((i+1)\*(int)(DD/DT)).Y[0]));

Table[i][5].setText(Double.toString(Database.get((i+1)\*(int)(DD/DT)).Y[4]));

Table[i][6].setText(Double.toString(Database.get((i+1)\*(int)(DD/DT)).NY));

}

SB.setMaximum((int)(TF/DD)-5);

SB.setValue(1);

SB.addAdjustmentListener(new TableScroll());

CreateGraphicX(max[1], min[1], 820, 60, 710, 350, 0);

CreateGraphicX(max[0], min[0], 820, 440, 710, 350, 4);

}

public void SHUM(int i) {

double SUMHR = 0;

for(int N = 1; N <=12; N++) {

SUMHR += Math.random();

}

hi = (SUMHR - 6)/6;

}

public void SAU(int i) {

if (i != 0)

Sigma=Kh\*(Database.get(i).Y[4]-Hz)+Kh\_D\*Database.get(i-1).X[4]+kwz\*Database.get(i).Y[1];

else

Sigma = Kh\*(-Hz);

if (Sigma > 10)

Sigma = 10;

else if (Sigma < -10) {

Sigma = -10;

}

Database.get(i).DV = Sigma;

//Database.get(i).DV = 0;

AV = Database.get(i).Y[3]+Database.get(i).Y[5]/C6;

}

public void DIN(int i) {

Database.get(i).X[0] = Database.get(i).Y[1]; //кутова швидкість тангажу

Database.get(i).X[1] = -C1\*Database.get(i).Y[1] - C2\*AV - C5\*Database.get(i).X[3] - C3\*Database.get(i).DV; //прискорення кутової швидкості тангажу

Database.get(i).X[2] = C4\*AV + C9\*Database.get(i).DV; //швидкість зміни кута нахилу траекторії

Database.get(i).X[3] = Database.get(i).X[0] - Database.get(i).X[2]; //швидкість зміни кута атаки

Database.get(i).X[4] = C6\*Database.get(i).Y[2]; //швидкість зміни висоти

Database.get(i).X[5] = Database.get(i).Y[6]+1.73\*Math.sqrt(Ry)\*Sigma\_Wy\*hi/Math.sqrt(DT); //прискорення вертикальної складової швидкості турбулентного вітру

Database.get(i).X[6] = -2\*Ry\*Database.get(i).Y[6]-Ry\*Ry\*Database.get(i).Y[5]-2.46\*Ry\*Math.sqrt(Ry)\*Sigma\_Wy\*hi/Math.sqrt(DT); //переміжна змінна

Database.get(i).NY = C16 \* Database.get(i).X[2];

}

public void EILER(int i) {

double[] YN = new double[7];

for(int j = 0; j < 7; j++)

YN[j] = Database.get(i).Y[j] + Database.get(i).X[j]\*DT;

Database.add(new Data(YN[0], YN[1], YN[2], YN[3], YN[4], YN[5], YN[6]));

}

public void CheckMaxMin(int j) {

if(Database.get(j).NY > max[1])

max[1] = Database.get(j).NY;

if(Database.get(j).NY < min[1])

min[1] = Database.get(j).NY;

if(Database.get(j).Y[4] > max[0])

max[0] = Database.get(j).Y[4];

if(Database.get(j).Y[4] < min[0])

min[0] = Database.get(j).Y[4];

}

public void paint (Graphics gr) {

gr.setColor(Color.lightGray);

gr.fillRect(0, 0, 1545, 830);

gr.setColor(Color.BLACK);

gr.drawLine(0, 550, 765, 550);

gr.drawLine(765, 0, 765, 830);

gr.setFont(new Font("TimesRoman", Font.PLAIN, 25));

gr.drawString("Таблиця даних", 280, 60);

gr.drawString("Графіки", 1100, 50);

gr.drawString("Розрахунок коефіцієнтів", 220, 600);

gr.setFont(new Font("TimesRoman", Font.PLAIN, 18));

gr.drawString("C1 =", 20, 650);

gr.drawString("C2 =", 200, 650);

gr.drawString("C3 =", 380, 650);

gr.drawString("C4 =", 560, 650);

gr.drawString("C5 =", 20, 700);

gr.drawString("C6 =", 200, 700);

gr.drawString("C9 =", 380, 700);

gr.drawString("C16 =", 550, 700);

gr.drawString("αбал =", 10, 750);

gr.drawString("Cyбал =", 180, 750);

gr.drawString("δбал =", 370, 750);

gr.drawString("mx =", 560, 750);

gr.setColor(Color.white);

gr.fillRect(30, 80, 680, 440);

}

public TextField CreatCTF(String value, int x, int y) {

TextField TF = new TextField(value);

TF.setBounds(x, y, 100, 25);

TF.setEditable(false);

TF.setFont(new Font("TimesRoman", Font.PLAIN, 18));

add(TF);

return TF;

}

public Button CreateBT(String value, int x, int y, int w, int h) {

Button BT = new Button(value);

BT.setBounds(x, y, w, h);

BT.setFont(new Font("TimesRoman", Font.PLAIN, 18));

add(BT);

return BT;

}

public void CreateGraphicX(double max, double min, int x, int y, int w, int h, int mas) {

int bub;

int bx, lx = x;

int by, ly = (int)map(Database.get(0).Y[mas], min, max, y+h, y);

if (mas == 0) {

ly = (int)map(Database.get(0).NY, min, max, y+h, y);

}

graph.setColor(Color.white);

graph.fillRect(x, y, w, h);

graph.setFont(new Font("TimesRoman", Font.PLAIN, 20));

double start, end, delta, mod = Math.abs(max - min);

if (mod > 10) {

delta = 5;

start = Math.ceil(min\*2/10)\*10/2;

end = Math.floor(max);

}

else if (mod > 7) {

delta = 2;

start = Math.ceil(min/2)\*2;

end = Math.floor(max/2)\*2;

}

else if (mod > 3) {

delta = 1;

start = Math.ceil(min);

end = Math.floor(max);

}

else if (mod > 1) {

delta = 0.5;

start = Math.ceil(min\*2)/2;

end = Math.floor(max\*2)/2;

}

else if (mod > 0.3) {

delta = 0.1;

start = Math.ceil(min\*10)/10;

end = Math.floor(max\*10)/10;

}

else if (mod > 0.15) {

delta = 0.05;

start = Math.ceil(min\*20)/20;

end = Math.floor(max\*20)/20;

}

else {

delta = 0.02;

start = Math.ceil(min\*500)/500;

end = Math.floor(max\*500)/500;

}

for (double i = start; i <= end; i+=delta) {

bub = (int)map(i, min, max, y+h, y);

if((int)Math.floor(i\*100) != 0) {

graph.setColor(Color.black);

if(i % 1 == 0)

graph.drawString(Integer.toString((int)Math.round(i\*100)/100), x-30, bub+10);

else if (((double)Math.round(i\*100)/100)\*10 % 1 == 0)

graph.drawString(Double.toString((double)Math.round(i\*100)/100), x-35, bub+10);

else

graph.drawString(Double.toString((double)Math.round(i\*100)/100), x-45, bub+10);

graph.setColor(Color.lightGray);

graph.drawLine(x, bub, x+w, bub);

}

}

for (int i = 5; i <= TF; i += TF/10) {

bub = (int)map(i, 0, TF, x, x+w);

graph.setColor(Color.black);

graph.drawString(Integer.toString(i), bub, (int)map(0, min, max, y+h, y));

graph.setColor(Color.lightGray);

graph.drawLine(bub, y, bub, y+h);

}

graph.setColor(Color.black);

graph.fillRect(x, y, 2, h);

bub = (int)map(0, min, max, y+h, y);

graph.fillRect(x, bub, w, 2);

graph.setFont(new Font("TimesRoman", Font.PLAIN, 30));

graph.drawString("^", x-6, y+20);

graph.setFont(new Font("TimesRoman", Font.PLAIN, 30));

graph.drawString(">", x+w-15, bub+11);

graph.setFont(new Font("TimesRoman", Font.PLAIN, 20));

graph.drawString("T", x+w/2, y+h+20);

switch(mas) {

case 0:

graph.drawString("ny", x-30, bub+10);

break;

case 4:

graph.drawString("H", x-20, bub+10);

break;

}

graph.setColor(Color.red);

for(double i = 2\*DT; i < TF-0.1; i+=DT) {

bx = (int)map(i, 0, TF, x, x+w);

if (mas == 0)

by = (int)map((Database.get((int)(i/DT)).NY), min, max, y+h, y);

else

by = (int)map((Database.get((int)(i/DT)).Y[mas]), min, max, y+h, y);

graph.drawLine(lx, ly, bx, by);

graph.drawLine(lx, ly, bx+1, by);

graph.drawLine(lx, ly, bx, by+1);

graph.drawLine(lx+1, ly, bx, by);

graph.drawLine(lx, ly+1, bx, by);

graph.drawLine(lx, ly, bx-1, by);

graph.drawLine(lx, ly, bx, by-1);

graph.drawLine(lx-1, ly, bx, by);

graph.drawLine(lx, ly-1, bx, by);

lx = bx;

ly = by;

}

}

public double map(double input, double INmin, double INmax, double OUTmin, double OUTmax) {

return (input-INmin)\*(OUTmax-OUTmin)/(INmax-INmin)+OUTmin;

}

public class TableScroll implements AdjustmentListener {

public void adjustmentValueChanged(AdjustmentEvent e) {

int bub = SB.getValue();

for(int i = bub; i < bub+16; i++) {

Table[i-bub][0].setText(Double.toString((i)\*DD));

Table[i-bub][1].setText(Double.toString(Database.get((i)\*(int)(DD/DT)).Y[5]));

Table[i-bub][2].setText(Double.toString(Database.get((i)\*(int)(DD/DT)).DV));

Table[i-bub][3].setText(Double.toString(Database.get((i)\*(int)(DD/DT)).Y[3]));

Table[i-bub][4].setText(Double.toString(Database.get((i)\*(int)(DD/DT)).Y[0]));

Table[i-bub][5].setText(Double.toString(Database.get((i)\*(int)(DD/DT)).Y[4]));

Table[i-bub][6].setText(Double.toString(Database.get((i)\*(int)(DD/DT)).NY));

}

}

}

public static void main(String[] args) {

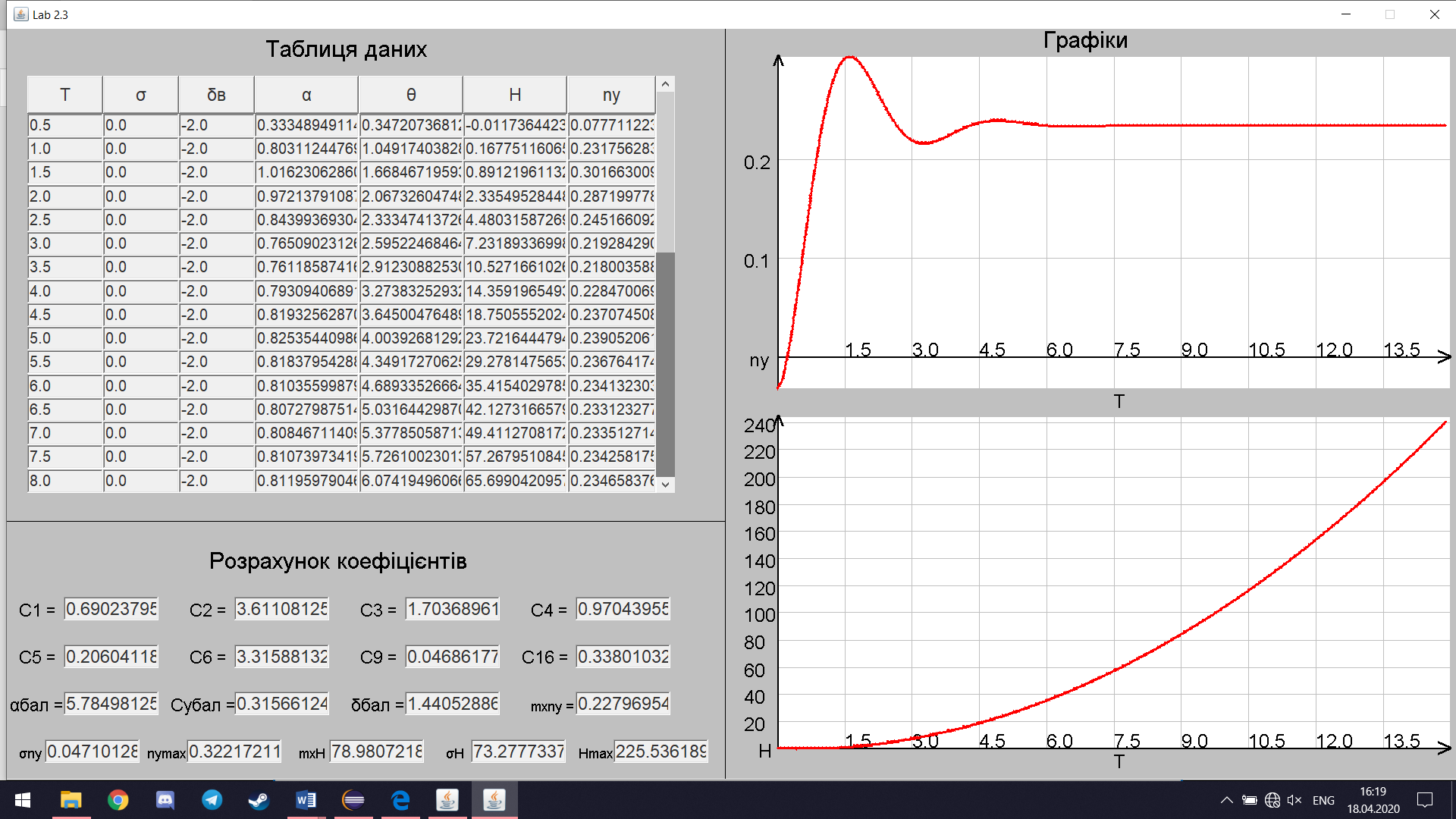
new Lab("Lab 2.3");

}

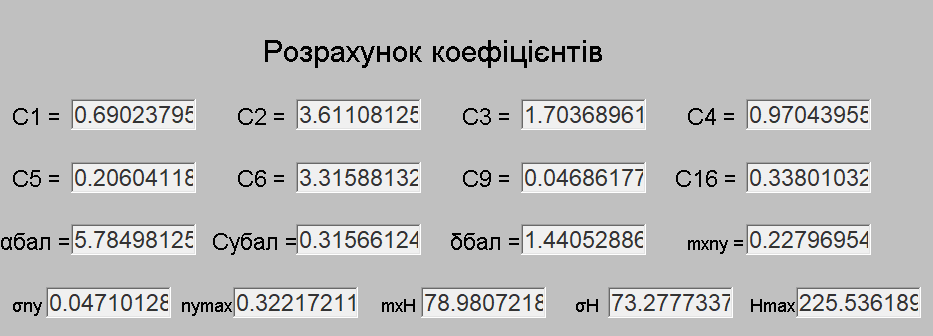
}

**Результати**

**Загальний вигляд програми**



**Чисельні результати досліджень**

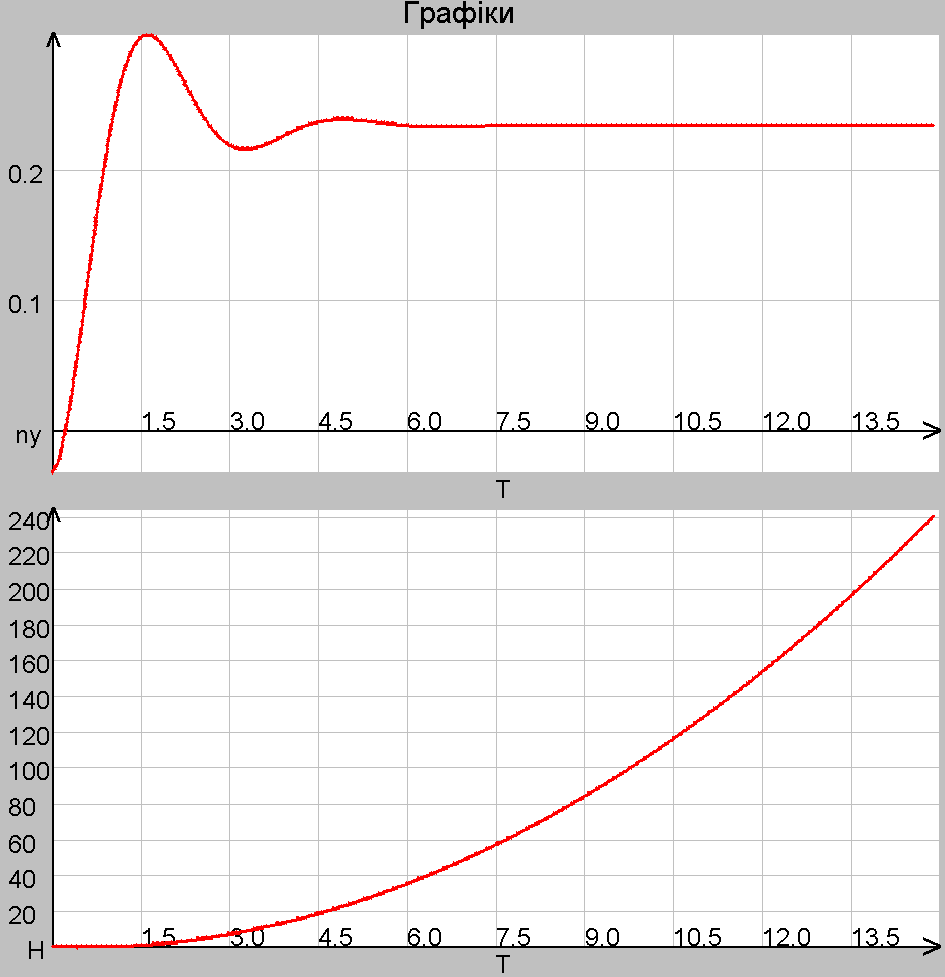


**Результати графічних досліджень**

Перший графік – графік зміни ny, другий графік - H.

1. Перевірка динамічної подібності моделі та реального об’єкта, через «дачу» руля висоти на величину = –20 в спокійній атмосфері.

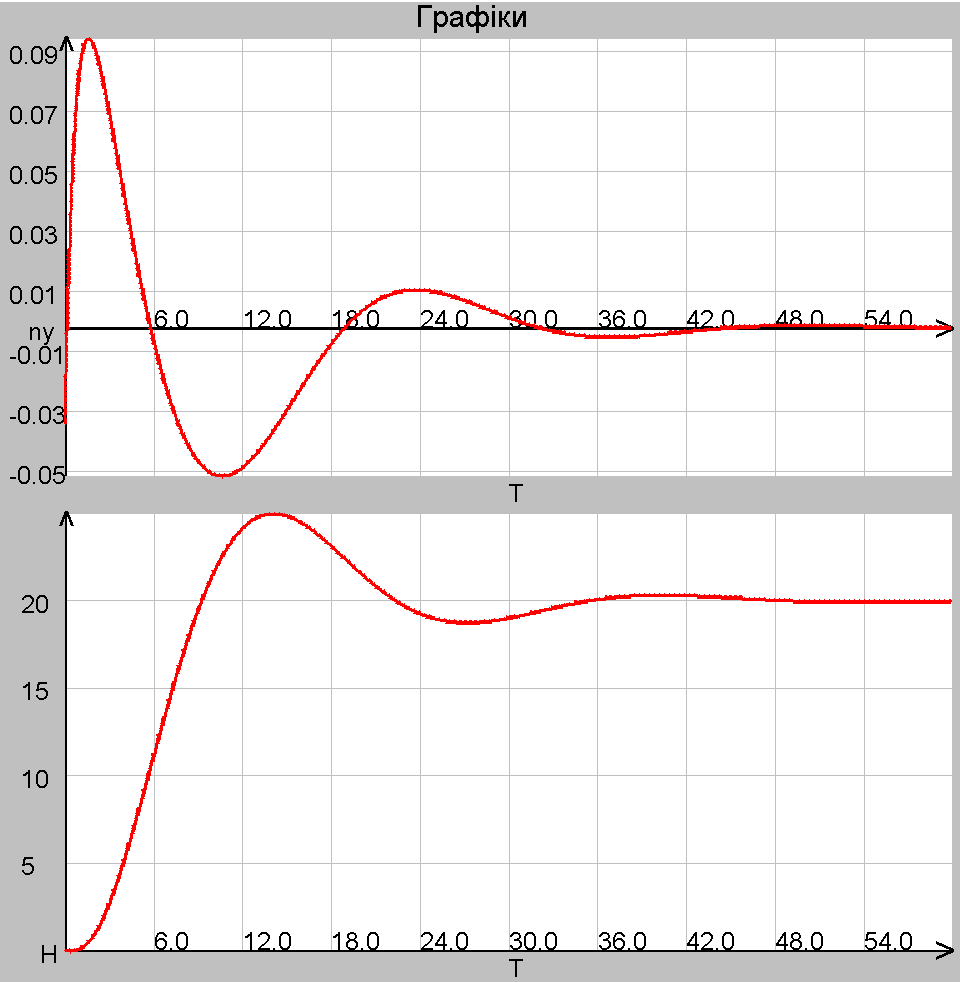


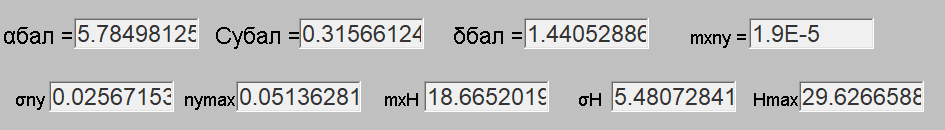


Статичні та динамічні характеристики повздовжньої керованості літака

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ,  град. | ,  од. | , град. | | *Тα* , с | | , град/од.пер. | | |
| Реальн.  об’єкт | Аналіт. розрах. | Реальн. об’єкт | Модель | Реальн. об’єкт | Аналіт. розрах. | Модель |
| 5.78 | 0.31 | 1.8 | 1.44 | 4-6 | 3.5 | -11 | -10.34 | -10 |

1. Дослідження якості автоматичної стабілізації заданої висоти *H*зад = 20 м в спокійній атмосфері.

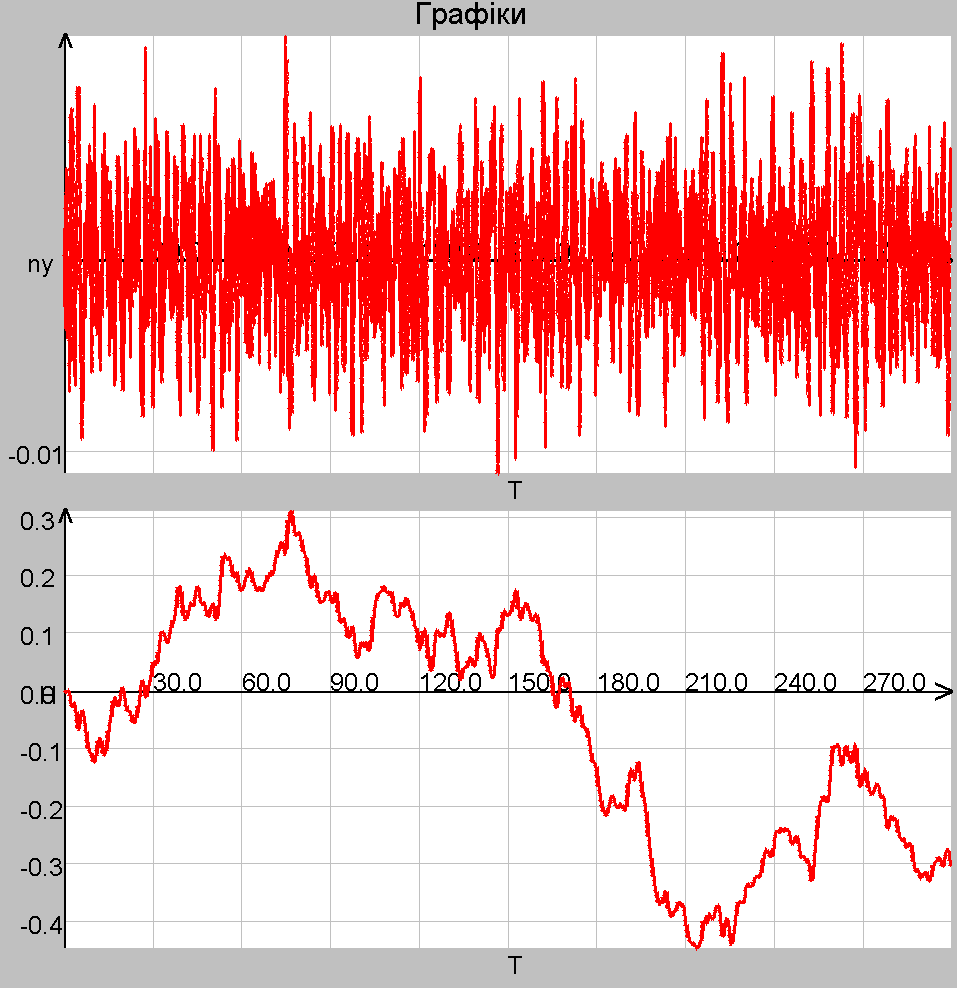


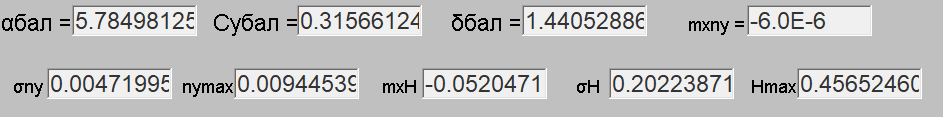


1. Дослідження збуреного руху «вільного» літака при дії вертикальної складової турбулентного вітру різної інтенсивності.

= 0,5 м/с

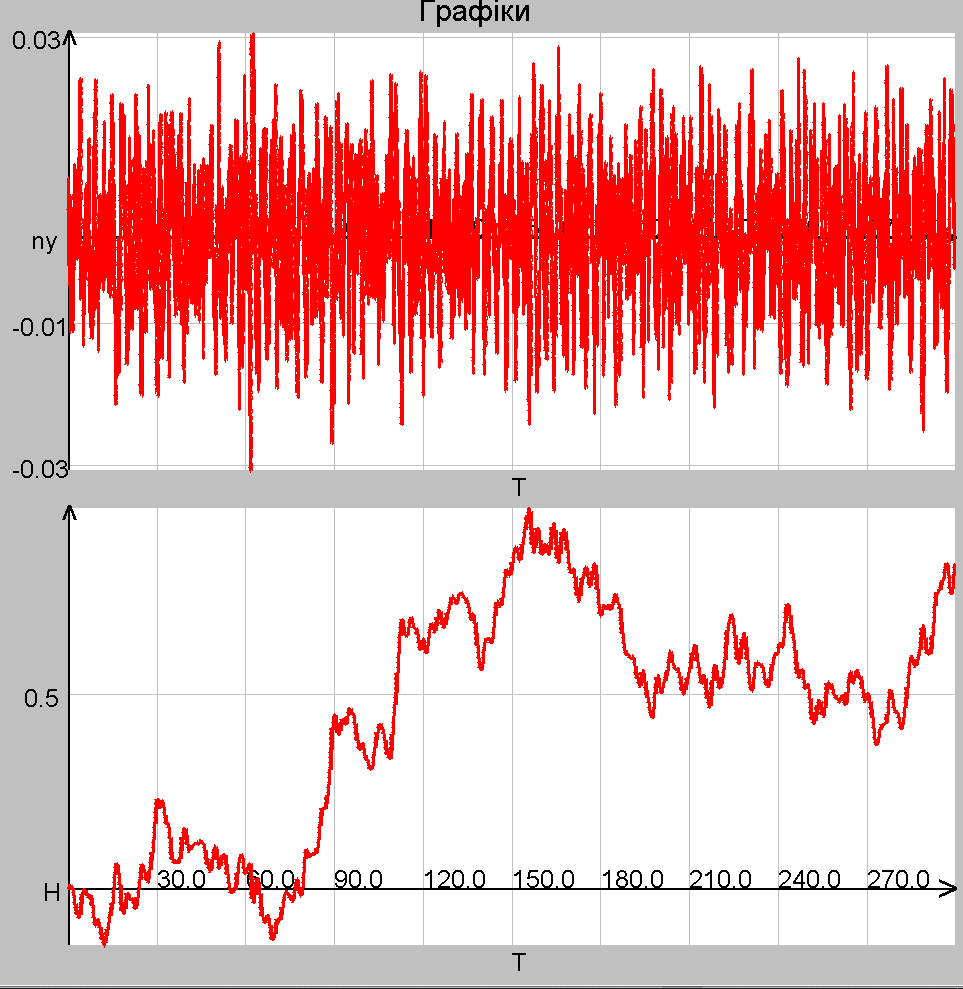


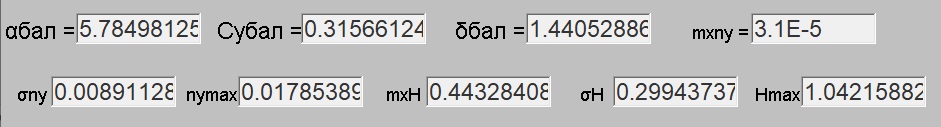




= 1,0 м/с

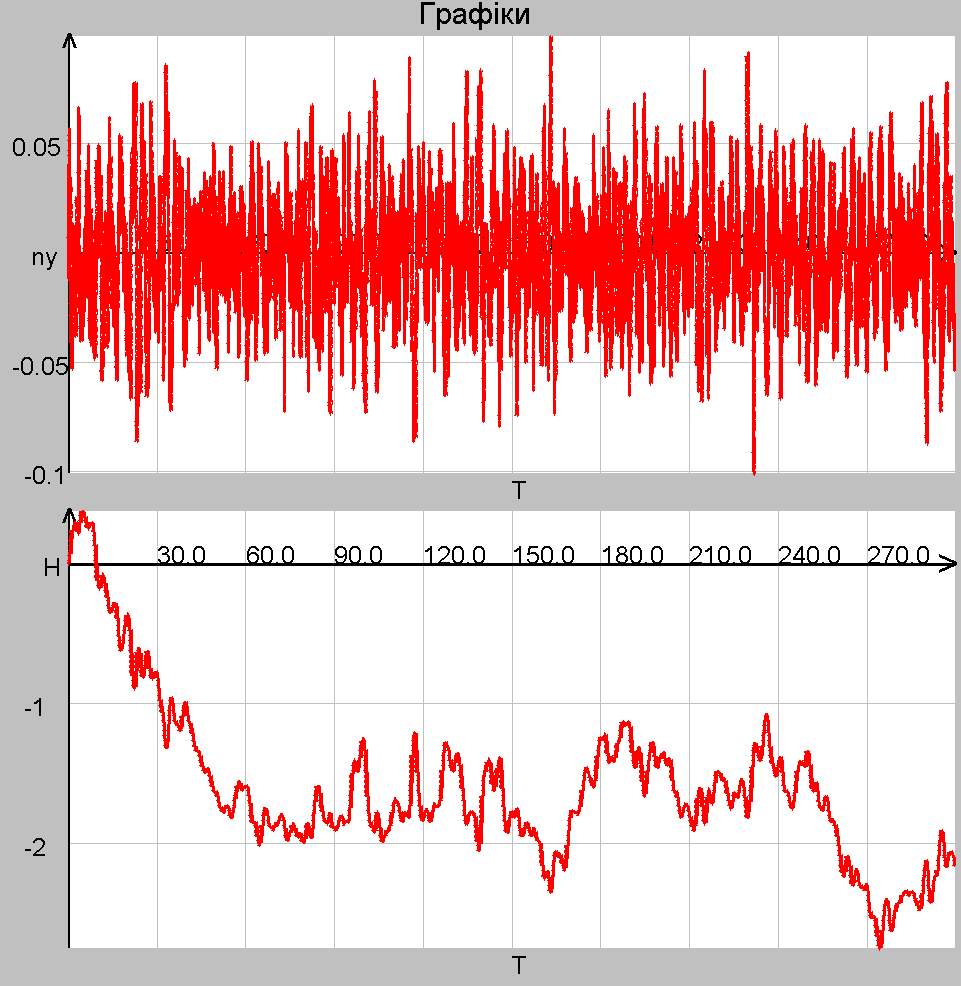


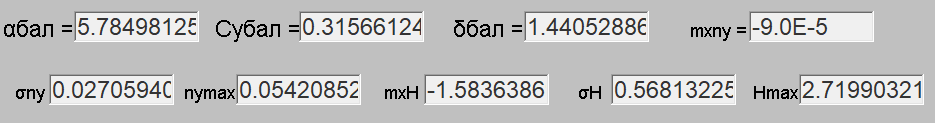




= 3,0 м/с



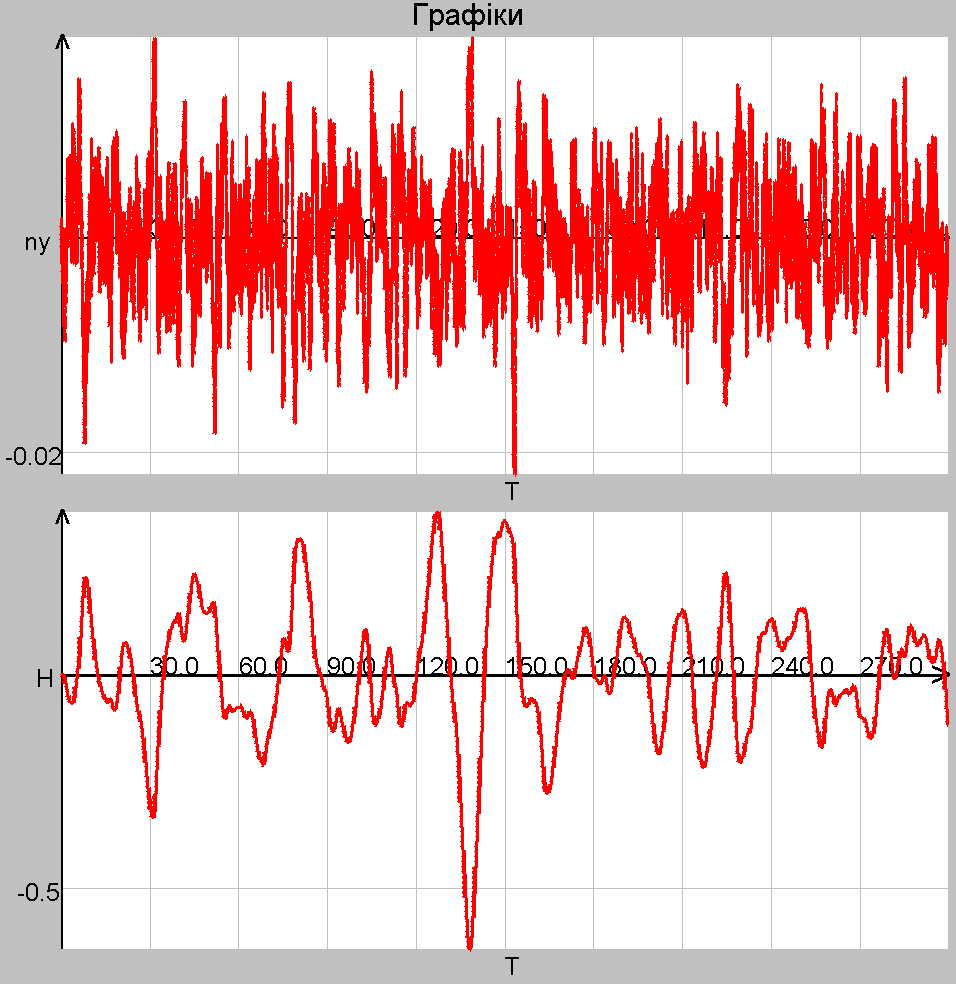


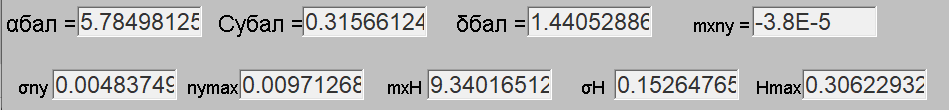


1. Дослідження збуреного руху літака в режимі автоматичної стабілізації заданої висоти *H*зад = 0 м при дії вертикальної складової турбулентного вітру різної інтенсивності.

= 0,5 м/с

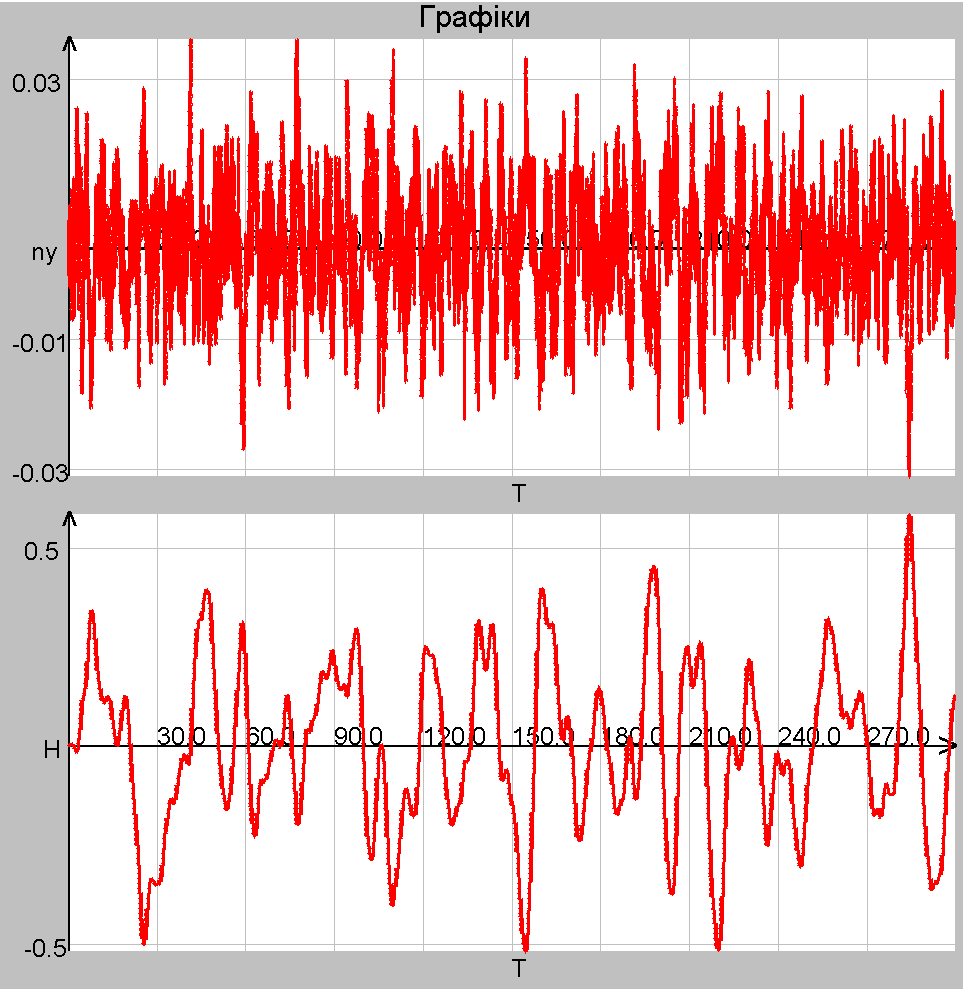


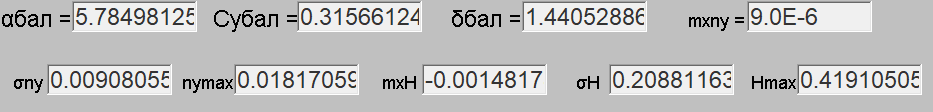




= 1,0 м/с

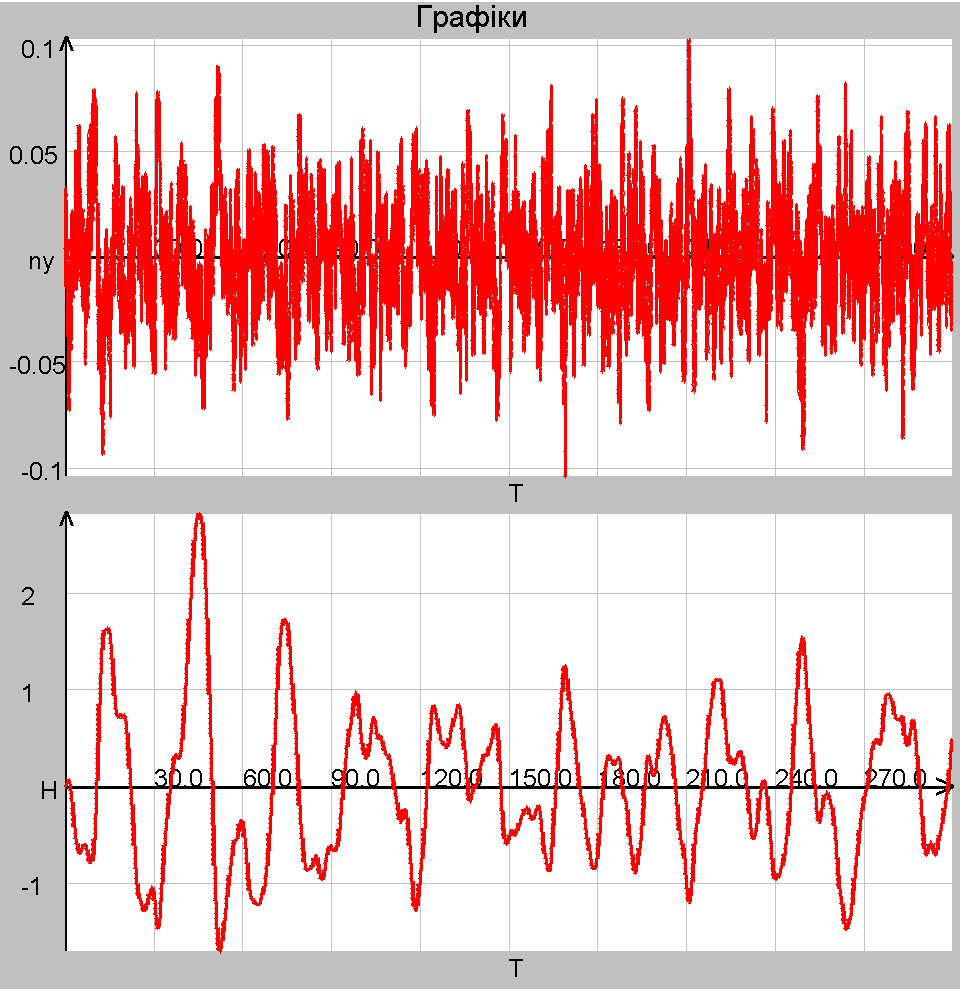


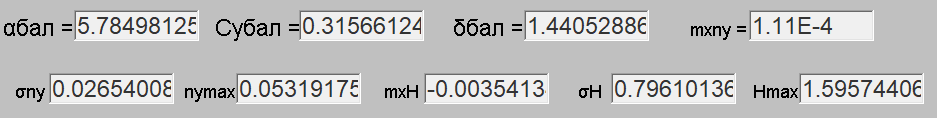




= 3,0 м/с







**Висновок про ступінь динамічної подібності розробленої цифрової математичної моделі і реального об'єкта.**

За результатами графічних та табличних досліджень можна стверджувати, що система є достатньо подібною, хоча і має незначні відхилення. Так відхилення руля висоти менше ніж у реальної моделі, як і період коливань.

**Висновок про якість стабілізації літака на заданій висоті в спокійній атмосфері.**

За результатами графічних досліджень в спокійній атмосфері система стабілізується правильно, з наступним утриманням висоти. Система є стійкою, але потребує певного часу на стабілізацію. Аналітичні розрахунки мають відхилення від графічних досліджень, але схожі.

**Висновок про якість стабілізації «вільного» та автоматично керованого руху літака на заданій висоті при дії атмосферної турбулентності різної інтенсивності.**

За результатами графічних досліджень при «вільному» керуванні система починає відхилятися від необхідної висоти, при ввімкненому САУ система намагається утримувати літак на необхідній висоті. Генератор «білого шуму» працює правильно, створюючи відхилення, як в додатному так і від’ємному напрямках. Чим більша інтенсивність вітру, тим менша стійкість САУ і тим більша зміна висоти. Аналітичні розрахунки мають відхилення особливо для вертикального перенавантаження. При ввімкненому САУ система набагато ефективніша при дії турбулентності, ніж при «вільному керувані».

**Кількісні оцінки інтенсивності «бовтанки» та інтенсивності турбулентності за величиною.**

За результатами графічних досліджень та аналітичних розрахунках можна стверджувати, що інтенсивності «бовтанки» та турбулентності у всіх випадках є слабкою. При «вільному» керуванні літака усі значення σny  та Δny для всіх інтенсивності менше 0.1. Аналогічні результати і при ввімкненому САУ.

**Висновок до лабораторної роботи:** було вивчено фізичну суть і характеристики атмосферної турбулентності та її вплив на динаміку польоту літака, особливості детермінованого та стохастичного моделювання атмосферної турбулентності. Досліджено вплив атмосферної турбулентності різної інтенсивності на якість автоматичної стабілізації заданої висоти польоту літака.