Міністерство освіти і науки України

Харківський національний університет ім. В. Н. Каразіна

Факультет комп’ютерних наук

Кафедра безпеки інформаційних систем і технологій

Лабораторна робота №3

*з навчальної дисципліни*

«Стеганографія»

Виконав:

Студент групи КБ-41

Кравченко Є.М.

Перевірив:

доцент

Нарєжній О.П.

Харків – 2020 р.

**Тема:** приховування даних в просторовій області нерухомих зображень на основі прямого розширення спектра.

**Мета роботи:** закріпити теоретичні знання за темою «Приховування даних у просторовій області нерухомих зображень на основі прямого розширення спектру», набути практичних вмінь та навичок щодо розробки стеганографічних систем, дослідити властивості стеганографічних методів, що засновані на низькорівневих властивостях зорової системи людини(ЗСЛ).

**1. Реалізація алгоритмів формування ансамблів ортогональних дискретних сигналів Уолша–Адамара та алгоритмів кодування інформаційних бітів даних складними дискретними сигналами**

Реалізуємо алгоритм формування матриць Адамара. Для цього скористаємося такою процедурою:

**public static int**[][] generateHadamardMatrix(**int** n) {  
 **int**[][] hadamard = **new int**[n][n];  
 hadamard[0][0] = 1;  
 **for** (**int** k = 1; k < n; k += k) {  
 **for** (**int** i = 0; i < k; i++) {  
 **for** (**int** j = 0; j < k; j++) {  
 hadamard[i + k][j] = hadamard[i][j];  
 hadamard[i][j + k] = hadamard[i][j];  
 hadamard[i + k][j + k] = -hadamard[i][j];  
 }  
 }  
 }  
 **return** hadamard;  
}

Вбудовувати повідомлення в контейнер будемо по рядках (кілька бітів в один рядок контейнера). Кодування складними дискретними сигналами зробимо у такий спосіб

**private int**[][] sum(**int**[] mpb, **int** rows) {  
 **int**[][] sum = **new int**[rows][**k**];  
 **for** (**int** i = 0; i < rows; i++) {  
 **int**[] a = **new int**[**arrayFunction**.**length**];  
 **for** (**int** j = 0; j < **k**; j++) {  
 a = sum1(a, multiply(**arrayFunction**[j + 1], mpb[**k** \* i + j], **g**));  
 }  
 **if** (**k** >= 0) System.*arraycopy*(a, 0, sum[i], 0, **k**);  
 }  
 **return** sum;  
}  
  
**private int**[] sum1(**int**[] a, **int**[] multiply) {  
 **int**[] res = **new int**[a.**length**];  
 **for** (**int** i = 0; i < a.**length**; i++) {  
 res[i] = a[i] + multiply[i];  
 }  
 **return** res;  
}  
  
**private int**[] multiply(**int**[] arrayFunction, **int** m, **int** g) {  
 **int**[] a = **new int**[arrayFunction.**length**];  
 **for** (**int** i = 0; i < a.**length**; i++) {  
 a[i] = arrayFunction[i] \* m \* g;  
 }  
 **return** a;  
}

**2. Реалізація алгоритмів приховування та вилучення даних шляхом прямого розширення спектрів із використанням ортогональних дискретних сигналів**

Реалізація алгоритма вбудовування інформаційних даних в просторову сферу зображення на основі прямого розширення спектра з використанням ортогональних дискретних сигналівУолша–Адамара:

@Override  
**public** BufferedImage encode(BufferedImage bufferedImage, String message) **throws** ThroughputException {  
 String mb = toBinaryString(message);  
 **int**[] mpb = mbToMbp(mb);  
 *//System.out.println(mb);* **int** n = (**int**) *log2*(Math.*min*(bufferedImage.getHeight(), bufferedImage.getWidth()));  
 **hm** = *generateHadamardMatrix*(n);  
 **int**[][] sum = sum(mpb, **k**);  
 *//System.out.println(Arrays.deepToString(sum));* **for** (**int** i = 0; i < **hm**.**length**; i++) {  
 **for** (**int** j = 0; j < **hm**.**length**; j++) {  
 Color color = **new** Color(bufferedImage.getRGB(i, j));  
 **int** newRed = color.getRed() + sum[i][j];  
 **if** (newRed > 255) newRed = 255;  
 **if** (newRed < 0) newRed = 0;  
 bufferedImage.setRGB(i, j, **new** Color(newRed, color.getGreen(), color.getBlue()).getRGB());  
 }  
 }  
 **return** bufferedImage;  
}  
  
**private int**[][] sum(**int**[] mpb, **int** k) {  
 **int**[][] sum = **new int**[**hm**.**length**][**hm**.**length**];  
 **for** (**int** i = 0; i < **hm**.**length**; i++) {  
 **int**[] a = **new int**[**hm**.**length**];  
 **for** (**int** j = 0; j < k; j++) {  
 **int**[] arr = **new int**[**hm**.**length**];  
 System.*arraycopy*(**hm**[j + 1], 0, arr, 0, **hm**.**length**);  
 **for** (**int** l = 0; l < arr.**length**; l++) {  
 arr[l] \*= (((i \* k + j) < mpb.**length**) ? mpb[i \* k + j] \* **g** : 0);  
 a[l] += arr[l];  
 }  
 }  
 System.*arraycopy*(a, 0, sum[i], 0, a.**length**);  
 }  
 **return** sum;  
}

Реалізація алгоритма вилучення інформаційних даних з просторової області зображення на основі прямого розширення спектра з використанням ортогональних дискретних сигналів Уолша–Адамара.

@Override  
**public** String decode(BufferedImage bufferedImage) {  
 **int** n = (**int**) *log2*(Math.*min*(bufferedImage.getHeight(), bufferedImage.getWidth()));  
 **hm** = *generateHadamardMatrix*(n);  
 StringBuilder mb = **new** StringBuilder();  
 **int**[][] arrayString = **new int**[**hm**.**length**][**hm**.**length**];  
 **int**[] a = **new int**[**hm**.**length**];  
 **for** (**int** i = 0; i < **hm**.**length**; i++) {  
 **for** (**int** j = 0; j < **hm**.**length**; j++) {  
 Color color = **new** Color(bufferedImage.getRGB(i, j));  
 a[j] = color.getRed();  
 }  
 System.*arraycopy*(a, 0, arrayString[i], 0, **hm**.**length**);  
 }  
 **for** (**int** i = 0; i < **hm**.**length**; i++) {  
 **for** (**int** j = 0; j < **k**; j++) {  
 mb.append(multString(arrayString[i], **hm**[j + 1]) > 0 ? **'1'** : **'0'**);  
 }  
 }  
 **byte**[] result = **new byte**[**hm**.**length** \* **k** / 8];  
 **for** (**int** i = 0, k = 0; i + 8 < mb.length(); i += 8) {  
 **byte** b = (**byte**) (**int**) Integer.*valueOf*(mb.substring(i, i + 8), 2);  
 result[k++] = b;  
 }  
 **return new** String(result, StandardCharsets.***UTF\_8***);  
}  
  
**private int** multString(**int**[] a, **int**[] b) {  
 **int** res = 0;  
 **for** (**int** i = 0; i < a.**length**; i++) {  
 res += a[i] \* b[i];  
 }  
 **return** res;  
}

**3. Проведення експериментальних досліджень ймовірносних властивостей реалізованого методу, отримання емпіричних залежностей ймовірності правильного вилучення даних та частки внесених при цьому похибок до контейнера-зображення**

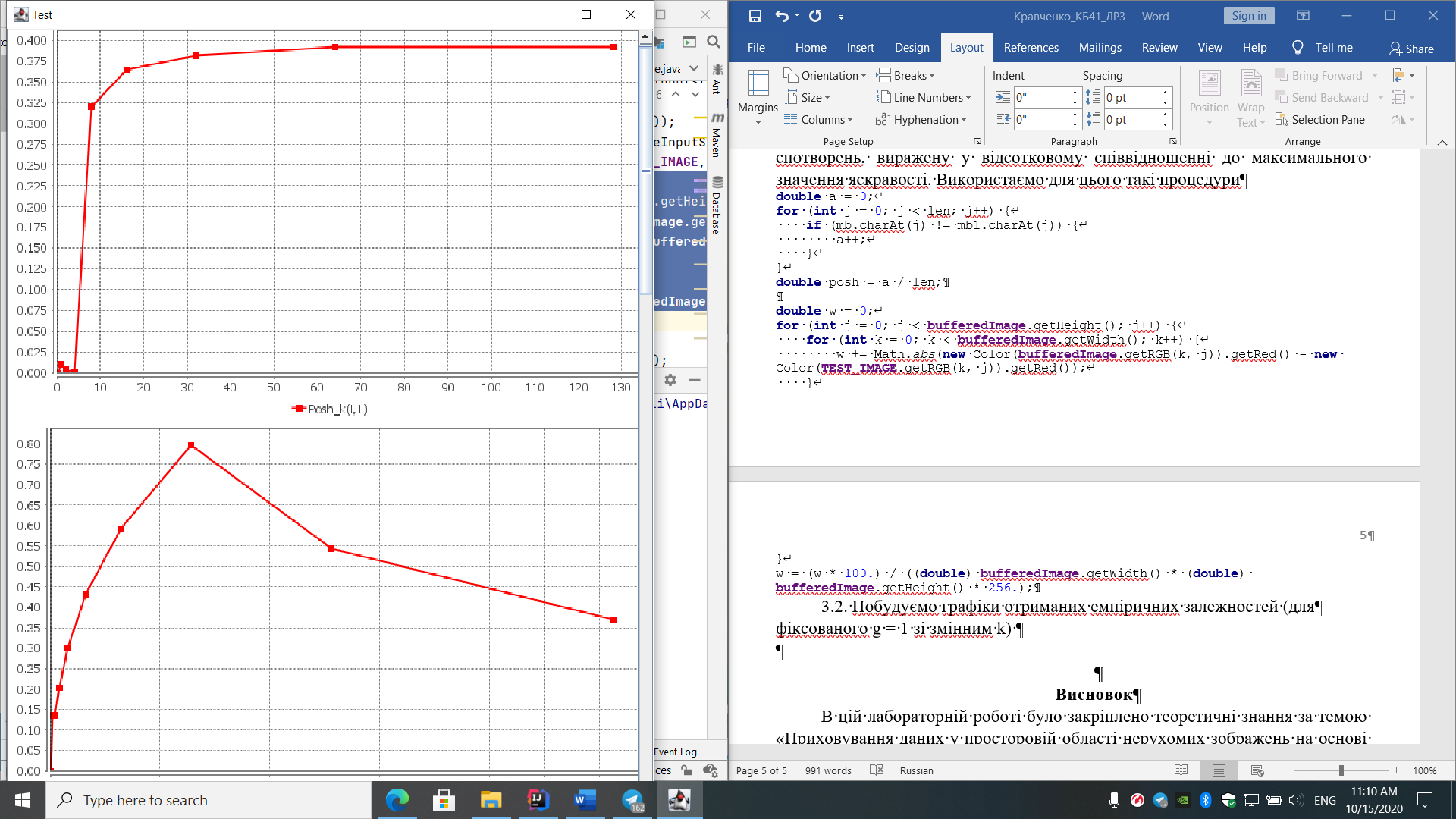
3.1. Проведемо оцінку ймовірності правильного вилучення повідомлення і обсягу внесених спотворень як від пропускної здатності стеганоканалу (задається величиною k), так і від коефіцієнта посилення g. Першу емпіричну залежність побудуємо у такий спосіб. Зафіксуємо g = 1, і за цього значення будемо послідовно збільшувати величину k. Для кожного значення k розрахуємо частоту Posh помилково вилучених інформаційних бітів. Одночасно будемо розраховувати усереднену величину w внесених спотворень, виражену у відсотковому співвідношенні до максимального значення яскравості. Використаємо для цього такі процедури

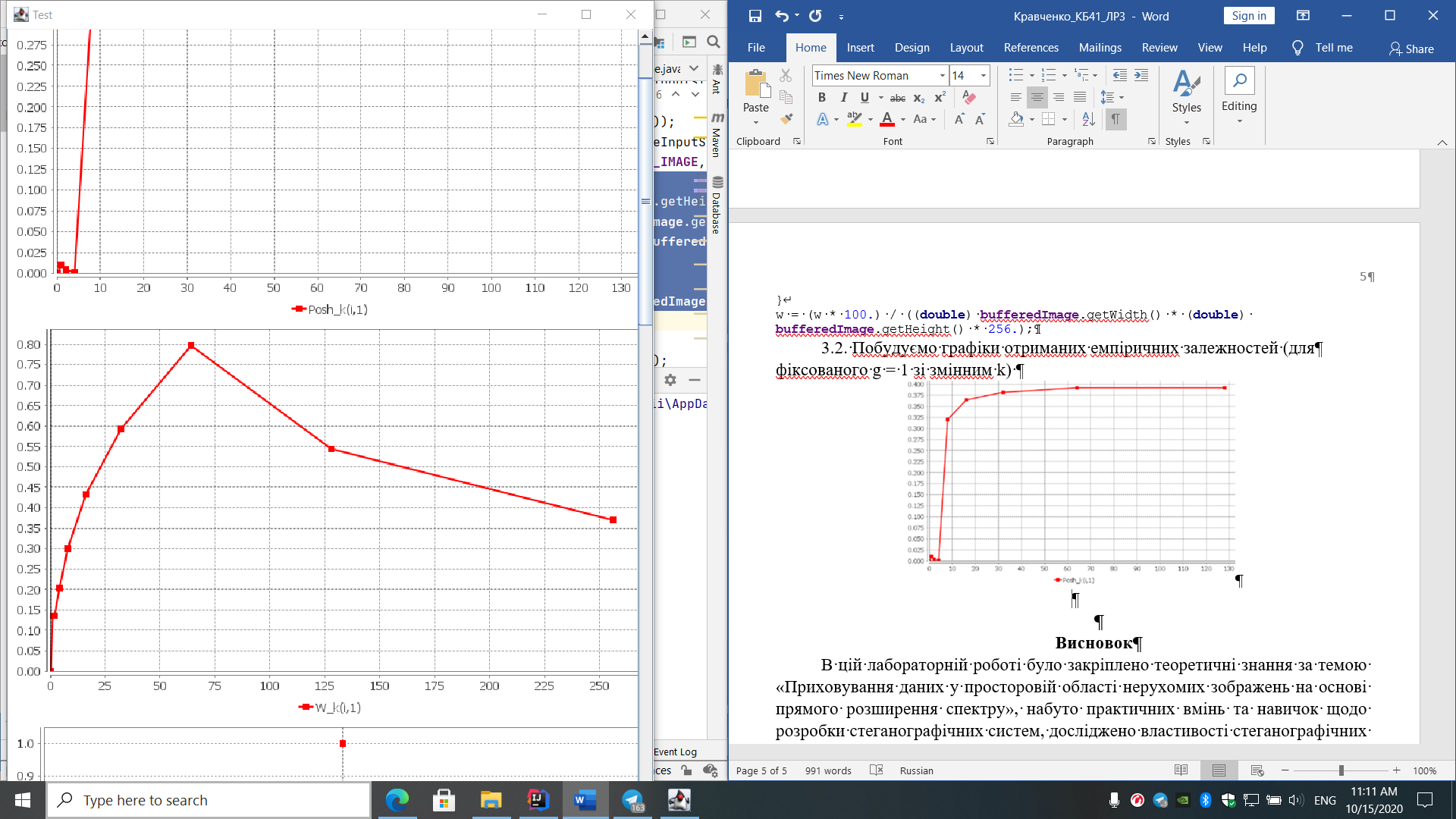
**double** a = 0;  
**for** (**int** j = 0; j < len; j++) {  
 **if** (mb.charAt(j) != mb1.charAt(j)) {  
 a++;  
 }  
}  
**double** posh = a / len;

**double** w = 0;  
**for** (**int** j = 0; j < **bufferedImage**.getHeight(); j++) {  
 **for** (**int** k = 0; k < **bufferedImage**.getWidth(); k++) {  
 w += Math.*abs*(**new** Color(**bufferedImage**.getRGB(k, j)).getRed() - **new** Color(**TEST\_IMAGE**.getRGB(k, j)).getRed());  
 }  
}  
w = (w \* 100.) / ((**double**) **bufferedImage**.getWidth() \* (**double**) **bufferedImage**.getHeight() \* 256.);

3.2. Побудуємо графіки отриманих емпіричних залежностей (для

фіксованого g = 1 зі змінним k)

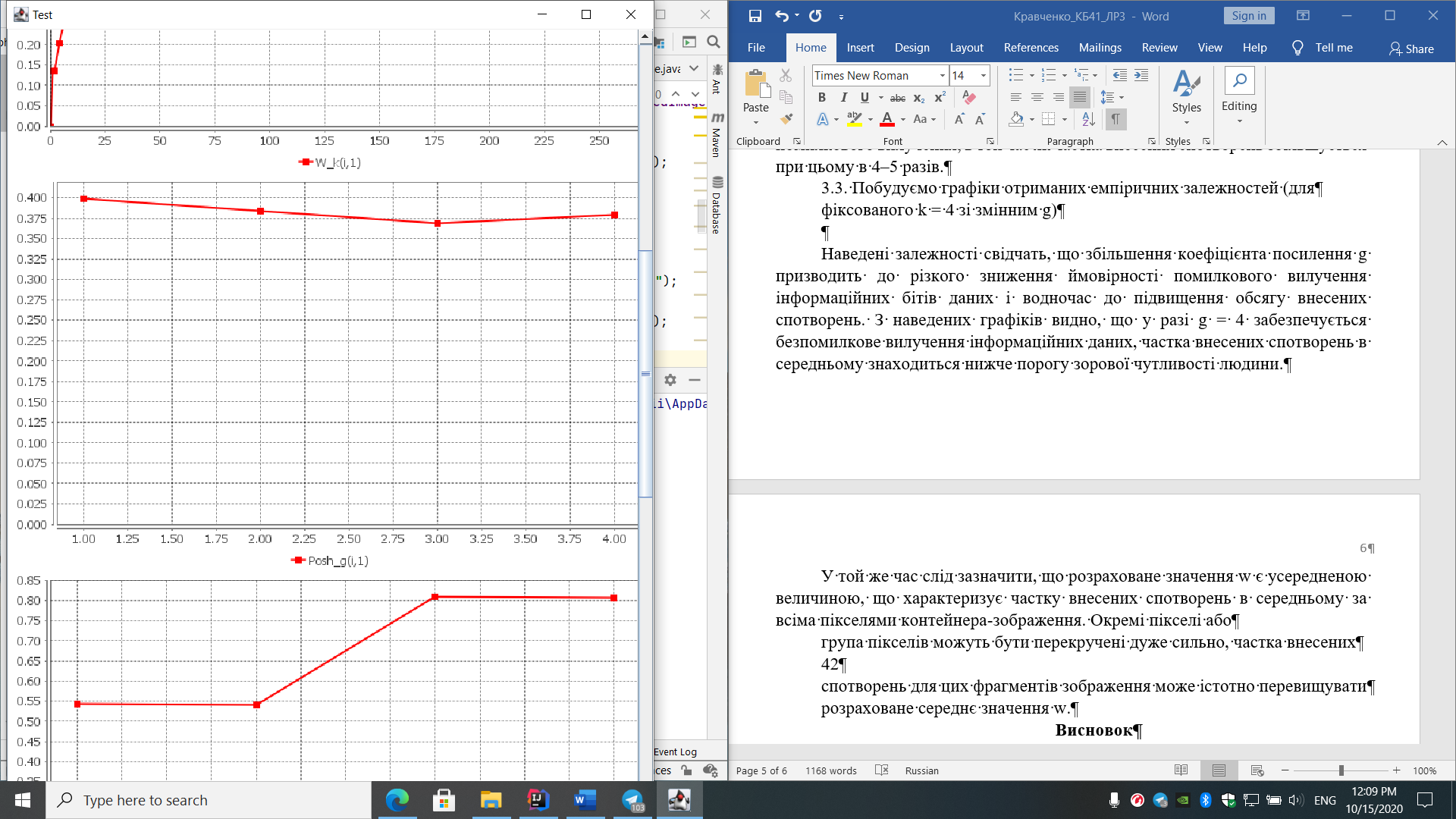


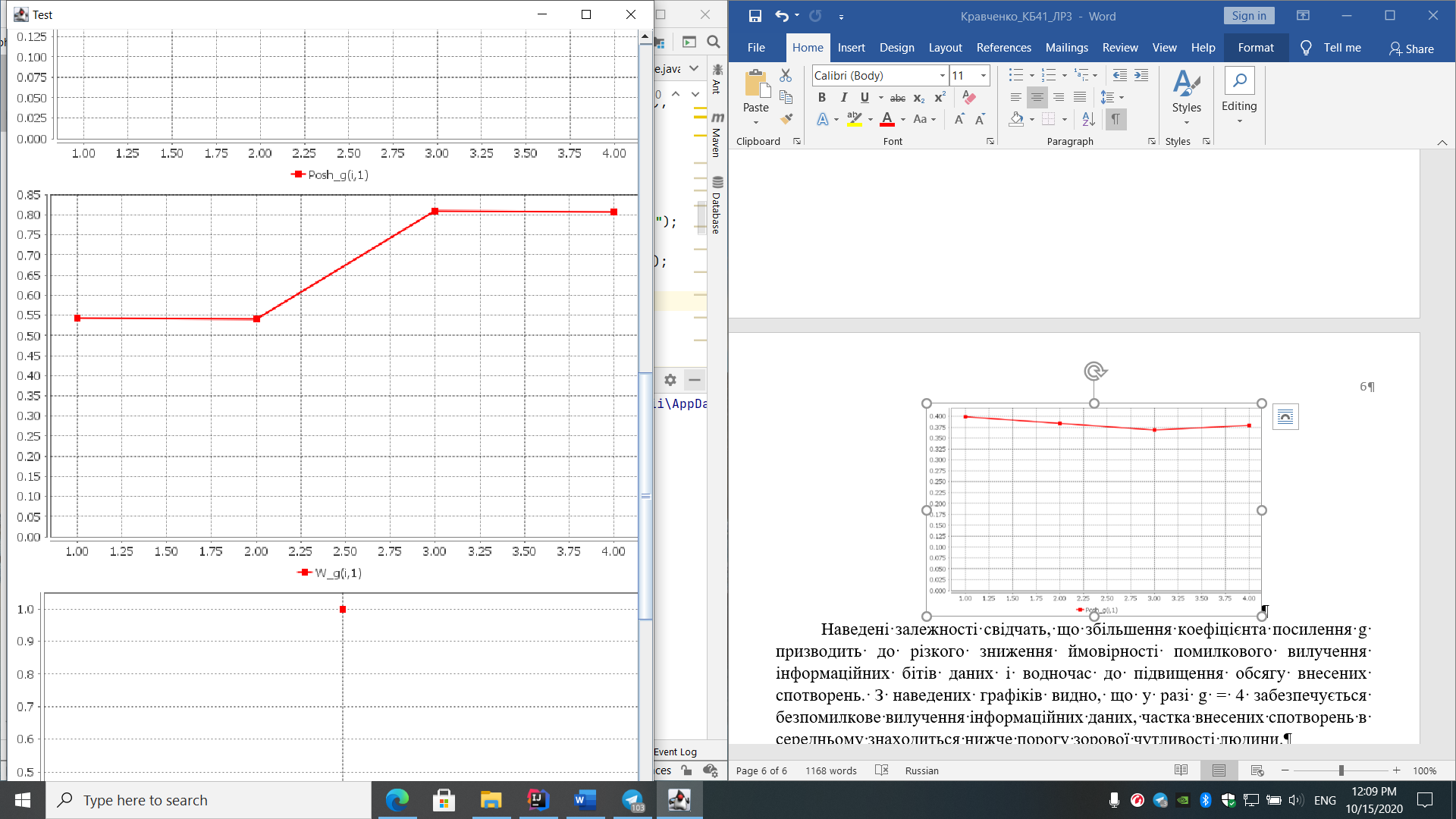


зрозуміло, що підвищення кількості вбудованих бітів даних призводить до збільшення як ймовірності помилкового вилучення даних, так і до підвищення частки внесених спотворень в контейнер-зображення. Слід зазначити, що збільшення кількості вбудованих бітів на один порядок (з 10 до 100 і вище) призводить до незначного (менше 0,05) збільшення ймовірності помилкового вилучення, в той час як частка внесених спотворень збільшується при цьому в 4–5 разів.

3.3. Побудуємо графіки отриманих емпіричних залежностей (для

фіксованого k = 4 зі змінним g)





Наведені залежності свідчать, що збільшення коефіцієнта посилення g призводить до різкого зниження ймовірності помилкового вилучення інформаційних бітів даних і водночас до підвищення обсягу внесених спотворень. З наведених графіків видно, що у разі g = 4 забезпечується безпомилкове вилучення інформаційних даних, частка внесених спотворень в середньому знаходиться нижче порогу зорової чутливості людини. У той же час слід зазначити, що розраховане значення w є усередненою величиною, що характеризує частку внесених спотворень в середньому за всіма пікселями контейнера-зображення. Окремі пікселі або група пікселів можуть бути перекручені дуже сильно, частка внесених спотворень для цих фрагментів зображення може істотно перевищувати розраховане середнє значення w.

**4. Реалізація алгоритмів формування ансамблів квазіортогональних дискретних сигналів та алгоритмів приховування та вилучення даних**

Алгоритм формування ансамблів квазіортогональних дискретних сигналів:

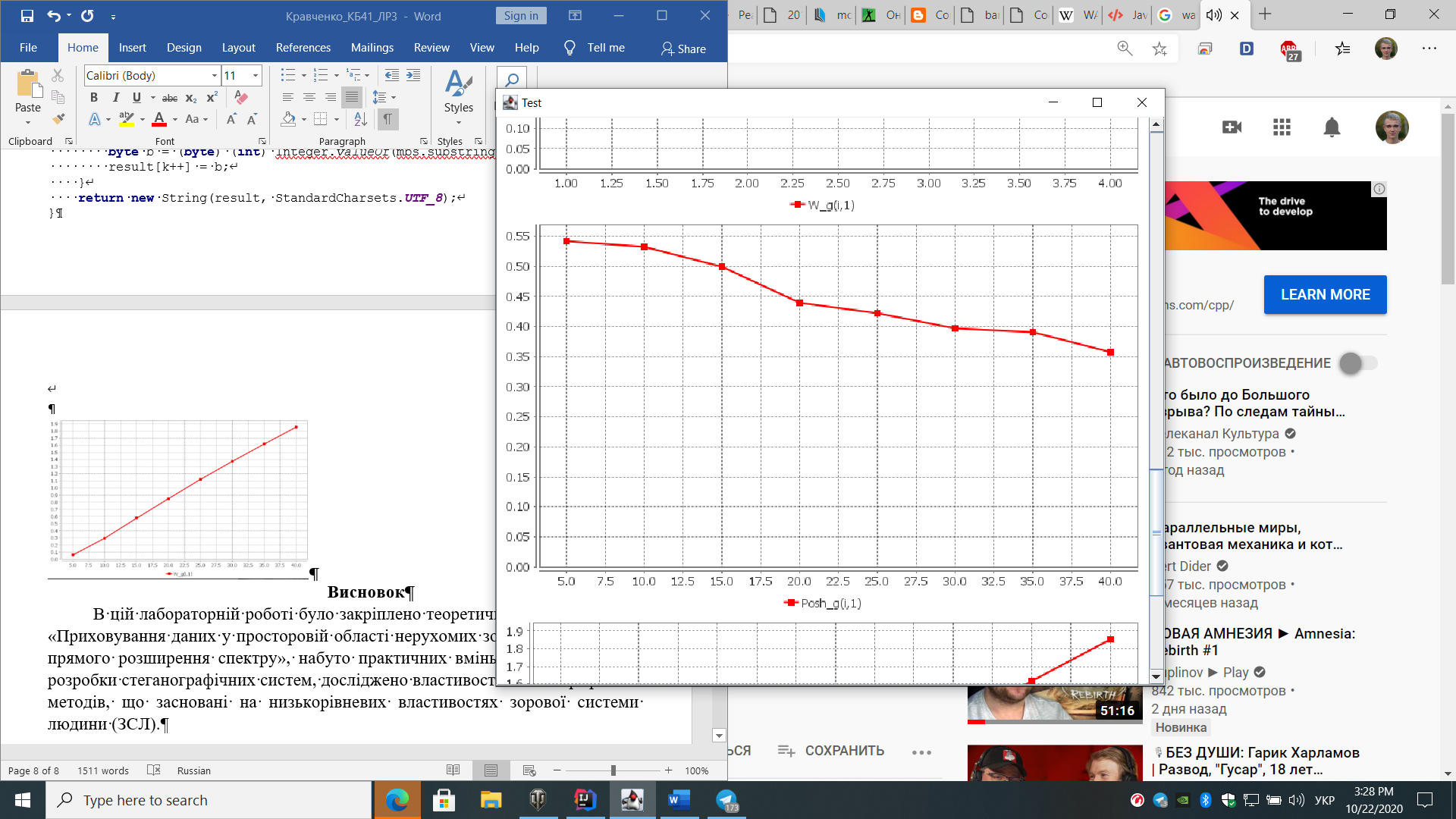
**private int**[][] arrayFunction() {  
 Random random = **new** Random(System.*nanoTime*());  
 **int**[][] res = **new int**[1024][**NUM**];  
 **for** (**int** i = 0; i < 1024; i++) {  
 **int**[] a = **new int**[**NUM**];  
 **for** (**int** j = 0; j < **NUM**; j++) {  
 **int** b = Math.*abs*(random.nextInt() % 2);  
 a[j] = b == 0 ? -1 : 1;  
 }  
 res[i] = a;  
 }  
 **return** res;  
}

Алгоритм приховування:

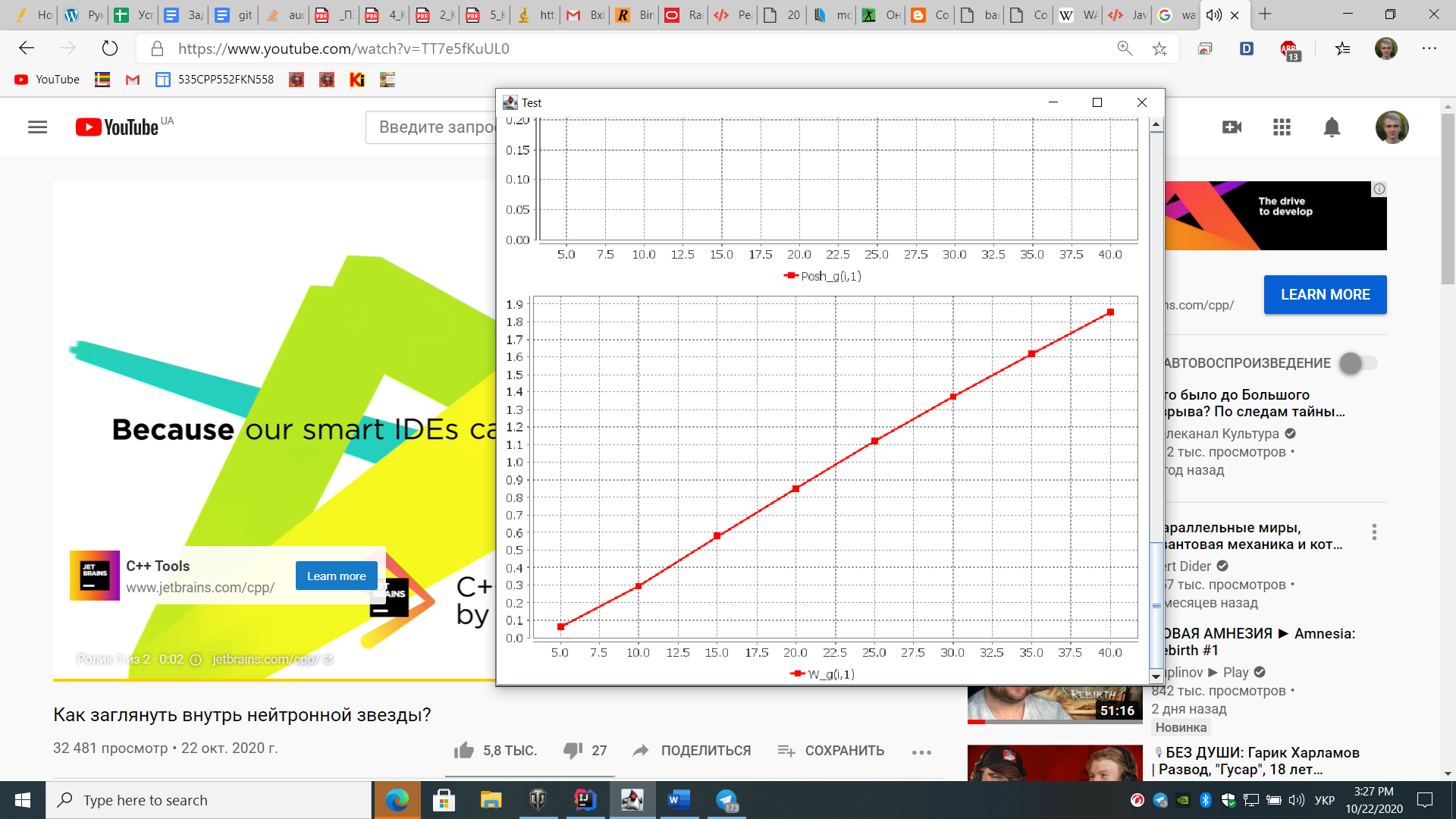
@Override  
**public** BufferedImage encode(BufferedImage bufferedImage, String message) **throws** ThroughputException {  
 String mb = toBinaryString(message);  
 **int**[] mb2 = mbToMb(mb);  
 **int**[] mb3 = **new int**[bufferedImage.getHeight() \* 10];  
 **int**[] md = **new int**[bufferedImage.getHeight()];  
 System.*arraycopy*(mb2, 0, mb3, 0, mb2.**length**);  
 **for** (**int** i = 0; i < mb3.**length** / 10; i++) {  
 **int** a = 0;  
 **for** (**int** j = 0; j < 10; j++) {  
 a += mb3[10 \* i + j] \* (**int**) Math.*pow*(2, j);  
 }  
 md[i] = a;  
 }  
 **for** (**int** i = 0; i < bufferedImage.getHeight(); i++) {  
 **for** (**int** j = 0; j < **NUM**; j++) {  
 Color color = **new** Color(bufferedImage.getRGB(j, i));  
 **int** newBlue = color.getBlue() + **g** \* **arrayFunction**[md[i]][j];  
 **if** (newBlue > 255) newBlue = 255;  
 **if** (newBlue < 0) newBlue = 0;  
 bufferedImage.setRGB(j, i, **new** Color(color.getRed(), color.getGreen(), newBlue).getRGB());  
 }  
 }  
 **return** bufferedImage;  
}  
Алгоритм вилучення:

@Override  
**public** String decode(BufferedImage bufferedImage) {  
 **arrayFunction** = readArrayFunction();  
 **int**[][] arrayString = **new int**[bufferedImage.getHeight()][**NUM**];  
 **for** (**int** i = 0; i < bufferedImage.getHeight(); i++) {  
 **for** (**int** j = 0; j < **NUM**; j++) {  
 arrayString[i][j] = **new** Color(bufferedImage.getRGB(j, i)).getBlue();  
 }  
 }  
 **int**[] md1 = **new int**[bufferedImage.getHeight()];  
 **for** (**int** i = 0; i < bufferedImage.getHeight(); i++) {  
 **int** a = 0;  
 **for** (**int** j = 0; j < 1024; j++) {  
 **if** (multString(arrayString[i], **arrayFunction**[j]) > a) {  
 a = multString(arrayString[i], **arrayFunction**[j]);  
 md1[i] = j;  
 }  
 }  
 }  
 **int**[] mb = **new int**[md1.**length** \* 10];  
 **for** (**int** i = 0; i < md1.**length**; i++) {  
 **int** x = md1[i];  
 **for** (**int** j = 0; j < 10; j++) {  
 mb[i \* 10 + j] = x % 2;  
 x /= 2;  
 }  
 }  
 StringBuilder mbs = **new** StringBuilder();  
 **for** (**int** j : mb) {  
 mbs.append(j);  
 }  
 **byte**[] result = **new byte**[(bufferedImage.getWidth() / 2 \* bufferedImage.getHeight()) / 8];  
 **for** (**int** i = 0, k = 0; i + 8 < mbs.length(); i += 8) {  
 **byte** b = (**byte**) (**int**) Integer.*valueOf*(mbs.substring(i, i + 8), 2);  
 result[k++] = b;  
 }  
 **return new** String(result, StandardCharsets.***UTF\_8***);  
}

Залежність внесених спотворень та ймовірності правильного вилучення



Залежність внесених спотворень від коефіцієнта посилення



**Висновок**

В цій лабораторній роботі було закріплено теоретичні знання за темою «Приховування даних у просторовій області нерухомих зображень на основі прямого розширення спектру», набуто практичних вмінь та навичок щодо розробки стеганографічних систем, досліджено властивості стеганографічних методів, що засновані на низькорівневих властивостях зорової системи людини (ЗСЛ).