**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ   
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО  Руководитель проекта, старший преподаватель факультета компьютерных наук департамента «Программная инженерия»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.В.Пантюхин  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2020 г. | |
| |  |  | | --- | --- | | Подп. и дата |  | | Инв. № дубл. |  | | Взам. Инв. № |  | | Подп. и дата |  | | Инв. № подл. | **RU.17701729.04.13-01 81 ТЗ 01-1** | | **Программа симуляции и визуализации работы радиолокатора Текст программы**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.04.13-01 81 ТЗ 01-1-ЛУ**  **Исполнитель**  Студентка группы БПИ 199  \_\_\_\_\_\_\_\_\_\_\_/Н.Д.Зубарева/  «\_\_\_»\_\_\_\_\_\_\_\_\_\_2020 г.    **Москва 2020** | | |  | |

**УТВЕРЖДЕНО**  
**RU.17701729.04.13-01 81 ТЗ 01-1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Подп. и дата |  | | Инв. № дубл. |  | | Взам. Инв. № |  | | Подп. и дата |  | | Инв. № подл. | **RU.17701729.04.13-01 81 ТЗ 01-1**  \_\_\_\_\_\_\_\_ | |  |  |

АННОТАЦИЯ

В данном программном документе приведен текст к «**Программе симуляции и визуализации работы радиолокатора»**. В разделе «Текст программы» находится текст программы, разделенный по классам. Программа разработана на языке C# 7.3 в среде разработки – Microsoft Visual Studio 2019 и использует функции, написанные на языке MATLAB версии 2020а.

Настоящий документ разработан в соответствии с требованиями:

1) ГОСТ 19.101-77 Виды программ и программных документов [1];

2) ГОСТ 19.102-77 Стадии разработки [2];

3) ГОСТ 19.103-77 Обозначения программ и программных документов [3];

4) ГОСТ 19.104-78 Основные надписи [4];

5) ГОСТ 19.105-78 Общие требования к программным документам [5];

6) ГОСТ 19.106-78 Требования к программным документам, выполненным

печатным способом [6];

7) ГОСТ 19.401-78 Текст программы. Требования к содержанию и

оформлению [7].

СОДЕРЖАНИЕ

[АННОТАЦИЯ 2](#_Toc39760791)

[СОДЕРЖАНИЕ 3](#_Toc39760792)

[1. ТЕКСТ ПРОГРАММЫ 4](#_Toc39760793)

[2.1. Код WinForms Coursework 4](#_Toc39760794)

[2.1.1 Код класса Homescreen.cs 4](#_Toc39760795)

[2.1.2 Код класса Input.cs 6](#_Toc39760796)

[2.1.3 Код класса Operations.cs 24](#_Toc39760797)

[2.2. Код библиотеки RadarLib 41](#_Toc39760798)

[2.2.1 Код класса RadarOperations.cs 41](#_Toc39760799)

[2.3. Код используемых MATLAB-функций 48](#_Toc39760800)

[2.3.1 Код функции CreateFiles.m 48](#_Toc39760801)

[2.3.2 Код функции CreateL.m 50](#_Toc39760802)

[2.3.3 Код функции CreatePicture.m 52](#_Toc39760803)

[2.3.4 Код функции getGauss.m 54](#_Toc39760804)

[2.3.5 Код функции getMatrNabl.m 54](#_Toc39760805)

[2.3.6 Код функции getPhase.m 54](#_Toc39760806)

[2.3.7 Код функции sinx.m 55](#_Toc39760807)

[2.3.8 Код функции Yinvert.m 55](#_Toc39760808)

[СПИСОК ЛИТЕРАТУРЫ 56](#_Toc39760809)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 57](#_Toc39760810)

1. ТЕКСТ ПРОГРАММЫ

Программа состоит из приложения Windows Forms, содержащего три класса: Homescreen, Input и Operations, которые реализуют взаимодействие пользователя с программой на различных стадиях работы, библиотеки RadarLib, содержащей класс RadarOperations, и прикрепленной в ссылки dll MatlabFuncs, сгенерированной автоматически на основе MATLAB-функций CreateFiles, CreateL и CreatePicture, которые, в свою очередь, используют MATLAB-функции getGauss, getPhase, getMatrNabl, sinx, Yinvert.

* 1. Код WinForms Coursework
     1. Код класса Homescreen.cs

1. using System;
2. using System.Collections.Generic;
3. using System.Drawing;
4. using System.Windows.Forms;
5. namespace Coursework
6. {
7. /// <summary>
8. /// Форма с начальным экраном приложения,
9. /// на ней отображается информация о программе,
10. /// авторе, и с нее можно перейти к работе с
11. /// наблюдениями.
12. /// </summary>
13. public partial class Homescreen : Form
14. {
15. /// <summary>
16. /// Экземпляр класса функций библиотеки с
17. /// функциями матлаб, создается в самом начале
18. /// работы, чтобы размер формы не перерасчитывался
19. /// в дальнейшем из-за форматирования.
20. /// </summary>
21. MatlabFuncs.Funcs funcs = new MatlabFuncs.Funcs();
22. /// <summary>
23. /// Поля для цветов, которые определяют внешний
24. /// вид формы.
25. /// </summary>
26. Color backColor = Color.FromArgb(46, 76, 86);
27. Color panelColor = Color.FromArgb(89, 139, 155);
28. Color hoverColor = Color.FromArgb(125, 167, 181);
29. Color textColor = Color.FromArgb(160, 238, 230);
30. /// <summary>
31. /// Конструктор формы, в котором инициализируются
32. /// компоненты формы и задается из внешний вид.
33. /// </summary>
34. public Homescreen()
35. {
36. InitializeComponent();
37. this.BackColor = backColor;
38. panel1.BackColor = panelColor;
39. foreach (Button b in GetAllControlsOfType<Button>(this))
40. {
41. b.FlatAppearance.MouseOverBackColor = hoverColor;
42. b.ForeColor = textColor;
43. }
44. foreach (Label l in GetAllControlsOfType<Label>(this))
45. {
46. l.ForeColor = textColor;
47. }
48. name.Visible = author.Visible = true;
49. info.Visible = false;
50. }
51. /// <summary>
52. /// Обработчик нажатия на кнопку перехода к
53. /// главному экрану, на котором отображается название
54. /// приложения и информация об авторе.
55. /// </summary>
56. /// <param name="sender"></param>
57. /// <param name="e"></param>
58. private void bMain\_Click(object sender, EventArgs e)
59. {
60. name.Visible = author.Visible = true;
61. info.Visible = false;
62. }
63. /// <summary>
64. /// Обработчик нажатия на кнопку перехода к информации,
65. /// делает видимым текст, поясняющий работу данного
66. /// приложения, и невидимым остальной текст.
67. /// </summary>
68. /// <param name="sender"></param>
69. /// <param name="e"></param>
70. private void bInfo\_Click(object sender, EventArgs e)
71. {
72. name.Visible = author.Visible = false;
73. info.Visible = true;
74. }
75. /// <summary>
76. /// Обработчик нажатия на кнопку перехода к работе,
77. /// скрывает эту форму и открывает форму для
78. /// задания параметров симуляции на основе этой формы.
79. /// </summary>
80. /// <param name="sender"></param>
81. /// <param name="e"></param>
82. private void bProceed\_Click(object sender, EventArgs e)
83. {
84. this.Hide();
85. new Input(this, funcs).Show();
86. }
87. /// <summary>
88. /// Обработчик нажатия на кнопку выхода, закрывает
89. /// данную форму и приложение.
90. /// </summary>
91. /// <param name="sender"></param>
92. /// <param name="e"></param>
93. private void bExit\_Click(object sender, EventArgs e)
94. {
95. this.Close();
96. Application.Exit();
97. }
98. /// <summary>
99. /// Метод, позволяющий получить все элементы
100. /// определенного типа на форме для их
101. /// более удобной настройки.
102. /// </summary>
103. /// <typeparam name="T"> Параметр для типа
104. /// желаемых элементов. </typeparam>
105. /// <param name="container"> Параметр для
106. /// контейнера, на котором требуется найти
107. /// элементы желаемого типа. </param>
108. /// <returns> Возвращает список элементов
109. /// контроля желаемого типа. </returns>
110. static private List<Control> GetAllControlsOfType<T>(Control container)
111. {
112. List<Control> controlList = new List<Control>();
113. foreach (Control c in container.Controls)
114. {
115. controlList.AddRange(GetAllControlsOfType<T>(c));
116. if (c is T)
117. {
118. controlList.Add(c);
119. }
120. }
121. return controlList;
122. }
123. }
124. }
     * 1. Код класса Input.cs
125. using System;
126. using System.Collections.Generic;
127. using System.Drawing;
128. using System.Windows.Forms;
129. using System.IO;
130. namespace Coursework
131. {
132. /// <summary>
133. /// Форма, на которой происходит задание параметров
134. /// симуляции посредством ввода в текстовые поля на
135. /// форме, либо загрузки из файла с последующей
136. /// возможностью сохранения параметров в выбранный
137. /// пользователем файл.
138. /// </summary>
139. public partial class Input : Form
140. {
141. /// <summary>
142. /// Поля для цветов, которые определяют внешний
143. /// вид формы.
144. /// </summary>
145. Color backColor = Color.FromArgb(46, 76, 86);
146. Color panelColor = Color.FromArgb(89, 139, 155);
147. Color hoverColor = Color.FromArgb(125, 167, 181);
148. Color textColor = Color.FromArgb(160, 238, 230);
149. /// <summary>
150. /// Поле для ссылки на предыдущую форму,
151. /// которая откроется в случае закрытия этой.
152. /// </summary>
153. Homescreen prevForm;
154. /// <summary>
155. /// Поле для ссылки на экземпляр класса
156. /// функций матлаб.
157. /// </summary>
158. MatlabFuncs.Funcs funcs;
159. /// <summary>
160. /// Поля для параметров симуляции.
161. /// </summary>
162. float Um, Vm, du, dv, u, v, a, Tc, up1, up2,
163. vp1, vp2, Ap1, Ap2, gamma, \_mu, H;
164. int N, Mu, Mv, pq;
165. /// <summary>
166. /// Конструктор формы, определающий внешний вид формы
167. /// и вызывающий метод, приводящий форму к дефолтному
168. /// состоянию.
169. /// </summary>
170. /// <param name="homescreen"> Параметр для ссылки на
171. /// предыдущую форму. </param>
172. /// <param name="funcs"> Параметр для экземпляра класса
173. /// функций матлаб. </param>
174. public Input(Homescreen homescreen, MatlabFuncs.Funcs funcs)
175. {
176. this.funcs = funcs;
177. prevForm = homescreen;
178. System.Globalization.CultureInfo.CurrentCulture
179. = new System.Globalization.CultureInfo("en-US", false);
180. this.BackColor = backColor;
181. InitializeComponent();
182. Begin();
183. }
184. /// <summary>
185. /// Метод, задающий внешний вид формы и значения
186. /// параметров симуляции их дефолтными значениями,
187. /// тут же происходит натсройка текста надписей
188. /// и подсказок для пользователя.
189. /// </summary>
190. void Begin()
191. {
192. panel.BackColor = panelColor;
193. foreach (Button b in GetAllControlsOfType<Button>(this))
194. {
195. b.FlatAppearance.MouseOverBackColor = hoverColor;
196. b.ForeColor = textColor;
197. }
198. foreach (Label l in GetAllControlsOfType<Label>(this))
199. {
200. l.ForeColor = textColor;
201. }
202. du = dv = Um = Vm = 0.1666667f;
203. u = v = vp1 = vp2 = gamma = 0;
204. a = 0.42f;
205. Tc = 20;
206. up1 = 0.03125f;
207. up2 = -0.03125f;
208. Ap1 = Ap2 = 300;
209. \_mu = 3000; H = 7;
210. N = 128;
211. Mu = Mv = 8;
212. pq = 576;
213. lN.Text = $"N = {N}";
214. lMu.Text = $"Mu = {Mv}";
215. lMv.Text = $"Mv = {Mu}";
216. lUm.Text = $"Um = {Um}";
217. lVm.Text = $"Vm = {Vm}";
218. ldu.Text = $"du = {dv}";
219. ldv.Text = $"dv = {dv}";
220. lu.Text = $"u = {u}";
221. lv.Text = $"v = {v}";
222. lTc.Text = $"Tc = {Tc}";
223. la.Text = $"a = {a}";
224. lup1.Text = $"up1 = {up1}";
225. lup2.Text = $"up2 = {up2}";
226. lvp1.Text = $"vp1 = {vp1}";
227. lvp2.Text = $"vp2 = {vp2}";
228. lAp1.Text = $"Ap1 = {Ap1}";
229. lAp2.Text = $"Ap2 = {Ap2}";
230. lgamma.Text = $"gamma = {gamma}";
231. l\_mu.Text = $"\_mu = {\_mu}";
232. lH.Text = $"H = {H}";
233. lpq.Text = $"pq = {pq}";
234. InN.Text = N.ToString();
235. InMu.Text = Mu.ToString();
236. InMv.Text = Mu.ToString();
237. InUm.Text = Um.ToString();
238. InVm.Text = Vm.ToString();
239. Indu.Text = dv.ToString();
240. Indv.Text = dv.ToString();
241. Inu.Text = u.ToString();
242. Inv.Text = v.ToString();
243. InTc.Text = Tc.ToString();
244. Ina.Text = a.ToString();
245. Inup1.Text = up1.ToString();
246. Inup2.Text = up2.ToString();
247. Invp1.Text = vp1.ToString();
248. Invp2.Text = vp2.ToString();
249. InAp1.Text = Ap1.ToString();
250. InAp2.Text = Ap2.ToString();
251. Ingamma.Text = gamma.ToString();
252. In\_mu.Text = \_mu.ToString();
253. InH.Text = H.ToString();
254. Inpq.Text = pq.ToString();
255. toolTip.SetToolTip(lN, "Количество векторов наблюдений," +
256. "\nпо которым формируется статистика." +
257. "\nЧисло должно быть целое и больше 0");
258. toolTip.SetToolTip(lMu, "Количество модулей, на которые" +
259. "\nразбивается антенная решетка по длине" +
260. "\nЧисло должно быть целое и больше 0");
261. toolTip.SetToolTip(lMv, "Количество модулей, на которые" +
262. "\nразбивается антенная решетка по ширине" +
263. "\nЧисло должно быть целое и больше 0");
264. toolTip.SetToolTip(lUm, "Ширина главного лепестка диаграммы" +
265. "\nнаправленности модуля по одному углу." +
266. "\nРекомендуемое значение: эффективная" +
267. "\nразрешающая способность модуля " +
268. "\nпо одному измерению - 1/6 радиана");
269. toolTip.SetToolTip(lVm, "Ширина главного лепестка диаграммы" +
270. "\nнаправленности модуля по другому углу." +
271. "\nРекомендуемое значение: эффективная" +
272. "\nразрешающая способность модуля " +
273. "\nпо одному измерению - 1/6 радиана");
274. toolTip.SetToolTip(ldu, "Сдвиг до центра " +
275. "просматриваемого сектора\nпо одному углу");
276. toolTip.SetToolTip(ldv, "Сдвиг до центра" +
277. " просматриваемого сектора\nпо другому углу");
278. toolTip.SetToolTip(lu, "Предполагаемое положение" +
279. " цели по одному углу,\nизменяется от -1/12 " +
280. "до 1/12 радиана с шагом 1/48 радиана");
281. toolTip.SetToolTip(lv, "Предполагаемое положение" +
282. " цели по другому углу,\nизменяется от -1/12" +
283. " до 1/12 радиана с шагом 1/48 радиана");
284. toolTip.SetToolTip(la, "Мощность приходящего сигнала." +
285. "\nЧисло должно быть больше 0");
286. toolTip.SetToolTip(lTc, "Начало прихода сигнала");
287. toolTip.SetToolTip(lvp1, "Положение помех по другому углу");
288. toolTip.SetToolTip(lvp2, "Положение помех по другому углу");
289. toolTip.SetToolTip(lup1, "Положение помех по одному углу");
290. toolTip.SetToolTip(lup2, "Положение помех по одному углу");
291. toolTip.SetToolTip(lAp1, "Мощность помех. " +
292. "Число должно быть больше 0");
293. toolTip.SetToolTip(lAp2, "Мощность помех. " +
294. "Число должно быть больше 0");
295. toolTip.SetToolTip(lgamma, "Параметр гамма");
296. toolTip.SetToolTip(l\_mu, "Параметр регуляризации корреляционной" +
297. " матрицы.\nЧисло должно быть больше 0");
298. toolTip.SetToolTip(lH, "Пороговое значение мощности сигнала" +
299. " для вывода точки.\nЧисло должно быть больше 0");
300. toolTip.SetToolTip(lpq, "Количество блоков в 36 областях," +
301. "\nпо которым составляется корреляционная матрица." +
302. "\nРекомендуемое значение:" +
303. "\n36 \* 2000 отсчетов по времени / N векторов выборки," +
304. "\nв противном случае размерности матриц будут конфликтующими.");
305. toolTip.SetToolTip(bGet, "Для того, чтобы загрузить параметры из файла,\n" +
306. "файл должен соответствовать определенным требованиям:" +
307. "\n-расширение .txt\n-параметры записаны каждый с новой строки без" +
308. "\nдругих разделителей\n-дробные числа записаны через ." +
309. "\n-параметры в файле идут в такой последовательности:" +
310. "\nN, Mu, Mv, Um, Vm, du, dv, u,\nv, a, Tc, up1, up2, vp1, vp2,\n" +
311. "Ap1, Ap2, gamma, mu, H, pq." +
312. "\nВ противном случае считывание будет некорректным.");
313. toolTip.SetToolTip(bExit, "Внимание!" +
314. "\nПри возвращении прогресс текущей сессии будет потерян." +
315. "\nУбедитесь в том, что вы сохранили все файлы," +
316. "\nкоторые не хотите потерять.");
317. }
318. /// <summary>
319. /// Обработчик нажатия на кнопку загрузки параметров
320. /// из файла. Появляется диалоговое окно для выбора
321. /// файла, затем вызывается метод получения параметров
322. /// из файла. Обрабатываются исключения, которые
323. /// могут возникнуть при чтении файла.
324. /// </summary>
325. /// <param name="sender"></param>
326. /// <param name="e"></param>
327. private void bGet\_Click(object sender, EventArgs e)
328. {
329. string inputPath = "";
330. if (openFileInput.ShowDialog()
331. == DialogResult.OK)
332. {
333. inputPath = openFileInput.FileName;
334. }
335. try
336. {
337. GetFromFile(File.ReadAllLines(inputPath));
338. }
339. catch (ArgumentException)
340. {
341. MessageBox.Show("Вы не выбрали файл.");
342. }
343. catch (FileNotFoundException)
344. {
345. MessageBox.Show("Файл не существует. " +
346. "Начните заново.");
347. }
348. catch (IOException)
349. {
350. MessageBox.Show("Ошибка в работе с файлом." +
351. " Начните заново.");
352. }
353. catch (UnauthorizedAccessException)
354. {
355. MessageBox.Show("Ошибка доступа к файлу:" +
356. " нет разрешения на доступ. Начните заново.");
357. }
358. catch (System.Security.SecurityException)
359. {
360. MessageBox.Show("Ошибка безопасности при " +
361. "работе с файлом. Начните заново.");
362. }
363. catch (Exception ex)
364. {
365. MessageBox.Show("Произошла ошибка: " + ex.Message);
366. }
367. }
368. /// <summary>
369. /// Метод для добавления параметров симуляции из
370. /// файла. Для считанных строк по одной на каждую
371. /// переменную в требуемом порядке их следования
372. /// вызывается метод изменения значения параметра
373. /// с проверкой корректности. Если данных не хватает,
374. /// оставшиеся параметры не меняются.
375. /// </summary>
376. /// <param name="inputStrings"> Параметр для строк,
377. /// считанных из файла. </param>
378. void GetFromFile(string[] inputStrings)
379. {
380. try
381. {
382. ChangeValueI(ref N, inputStrings[0], InN, lN, 1);
383. ChangeValueI(ref Mu, inputStrings[1], InMu, lMu, 1);
384. ChangeValueI(ref Mv, inputStrings[2], InMv, lMv, 1);
385. ChangeValueF(ref Um, inputStrings[3], InUm, lUm, 0 + float.Epsilon);
386. ChangeValueF(ref Vm, inputStrings[4], InVm, lVm, 0 + float.Epsilon);
387. ChangeValueF(ref du, inputStrings[5], Indu, ldu);
388. ChangeValueF(ref dv, inputStrings[6], Indv, ldv);
389. ChangeValueF(ref u, inputStrings[7], Inu, lu,
390. (float)(-1 / 12.0), (float)(1 / 12.0));
391. ChangeValueF(ref v, inputStrings[8], Inv, lv,
392. (float)(-1 / 12.0), (float)(1 / 12.0));
393. ChangeValueF(ref a, inputStrings[9], Ina, la);
394. ChangeValueF(ref Tc, inputStrings[10], InTc, lTc);
395. ChangeValueF(ref up1, inputStrings[11], Inup1, lup1);
396. ChangeValueF(ref up2, inputStrings[12], Inup2, lup2);
397. ChangeValueF(ref vp1, inputStrings[13], Invp1, lvp1);
398. ChangeValueF(ref vp2, inputStrings[14], Invp2, lvp2);
399. ChangeValueF(ref Ap1, inputStrings[15], InAp1, lAp1);
400. ChangeValueF(ref Ap2, inputStrings[16], InAp2, lAp2);
401. ChangeValueF(ref gamma, inputStrings[17], Ingamma, lgamma);
402. ChangeValueF(ref \_mu, inputStrings[18], In\_mu, l\_mu);
403. ChangeValueF(ref H, inputStrings[19], InH, lH, 0 + float.Epsilon);
404. ChangeValueI(ref pq, inputStrings[20], Inpq, lpq, 1);
405. }
406. catch (IndexOutOfRangeException)
407. {
408. MessageBox.Show("Информации в файле не хватает, но поля по максимуму заполнены");
409. }
410. finally { MessageBox.Show("Все корректные данные добавлены"); }
411. }
412. /// <summary>
413. /// Метод для изменения значения параметра
414. /// типа int. Вызывается метод проверки корректности
415. /// задаваемого значения, если данные корректны,
416. /// меняется поле параметра, надпись в связанном с
417. /// ним текстовом поле и поле ввода, иначе значение
418. /// не изменяется, а в поле ввода возвращается
419. /// предыдущее корректное значение.
420. /// </summary>
421. /// <param name="value"> Параметр для изменяемого
422. /// поля. </param>
423. /// <param name="newvalue"> Параметр для значения,
424. /// которое пользователь пытается задать. </param>
425. /// <param name="Invalue"> Параметр для текстового
426. /// поля, связанного с параметром симуляции. </param>
427. /// <param name="lvalue"> Параметр для лейбла,
428. /// связанного с параметром симуляции. </param>
429. /// <param name="min"> Минимальное значение,
430. /// допустимое для данного параметра. </param>
431. /// <param name="max"> Маскимальное значение,
432. /// допустимое для данного параметра. </param>
433. void ChangeValueI(ref int value, string newvalue,
434. TextBox Invalue, Label lvalue,
435. int min = int.MinValue, int max = int.MaxValue)
436. {
437. int num;
438. if (!CheckValidityFailedI(newvalue, out num, min, max))
439. {
440. value = num;
441. lvalue.Text = $"{lvalue.Name.Substring(1)} = {value}";
442. Invalue.Text = value.ToString();
443. }
444. else
445. {
446. Invalue.Text = value.ToString();
447. }
448. }
449. /// <summary>
450. /// Метод для изменения значения параметра
451. /// типа float. Вызывается метод проверки корректности
452. /// задаваемого значения, если данные корректны,
453. /// меняется поле параметра, надпись в связанном с
454. /// ним текстовом поле и поле ввода, иначе значение
455. /// не изменяется, а в поле ввода возвращается
456. /// предыдущее корректное значение.
457. /// </summary>
458. /// <param name="value"> Параметр для изменяемого
459. /// поля. </param>
460. /// <param name="newvalue"> Параметр для значения,
461. /// которое пользователь пытается задать. </param>
462. /// <param name="Invalue"> Параметр для текстового
463. /// поля, связанного с параметром симуляции. </param>
464. /// <param name="lvalue"> Параметр для лейбла,
465. /// связанного с параметром симуляции. </param>
466. /// <param name="min"> Минимальное значение,
467. /// допустимое для данного параметра. </param>
468. /// <param name="max"> Маскимальное значение,
469. /// допустимое для данного параметра. </param>
470. void ChangeValueF(ref float value, string newvalue,
471. TextBox Invalue, Label lvalue,
472. float min = float.MinValue, float max = float.MaxValue)
473. {
474. float num;
475. if (!CheckValidityFailedF(newvalue, out num, min, max))
476. {
477. value = num;
478. lvalue.Text = $"{lvalue.Name.Substring(1)} = {value}";
479. Invalue.Text = value.ToString();
480. }
481. else
482. {
483. Invalue.Text = value.ToString();
484. }
485. }
486. /// <summary>
487. /// Метод проверки корректности вводимых данных
488. /// для параметра типа float. Проверяется то, что
489. /// принимаемая на вход строка не пуста, что
490. /// она парсится в число требуемого типа, что
491. /// значение этого числа лежит в требуемых границах.
492. /// Если что-то из этого не выполняется,
493. /// возвращается булевое значение, говорящее о том,
494. /// что попытка изменения некорректна.
495. /// </summary>
496. /// <param name="newText"> Параметр для проверяемой
497. /// строки с значением, на которое пользователь
498. /// хочет изменить параметр. </param>
499. /// <param name="number"> Метод возвращает через
500. /// модификатор out число, полученное в результате
501. /// парсинга. </param>
502. /// <param name="min"> Минимальное значение,
503. /// возможное для параметра. </param>
504. /// <param name="max"> Максимальное значение,
505. /// возможное для параметра. </param>
506. /// <returns></returns>
507. public bool CheckValidityFailedF(string newText, out float number,
508. float min, float max)
509. {
510. number = 0;
511. if (string.IsNullOrEmpty(newText))
512. {
513. return true;
514. }
515. if (!float.TryParse(newText, out number))
516. {
517. MessageBox.Show("Вы не можете изменить значение " +
518. "ячейки на такое значение!");
519. return true;
520. }
521. if (number < min || number > max)
522. {
523. MessageBox.Show("Число находится вне " +
524. "границ, допустимых для этого параметра!");
525. return true;
526. }
527. return false;
528. }
529. /// <summary>
530. /// Метод проверки корректности вводимых данных
531. /// для параметра типа int. Проверяется то, что
532. /// принимаемая на вход строка не пуста, что
533. /// она парсится в число требуемого типа, что
534. /// значение этого числа лежит в требуемых границах.
535. /// Если что-то из этого не выполняется,
536. /// возвращается булевое значение, говорящее о том,
537. /// что попытка изменения некорректна.
538. /// </summary>
539. /// <param name="newText"> Параметр для проверяемой
540. /// строки с значением, на которое пользователь
541. /// хочет изменить параметр. </param>
542. /// <param name="number"> Метод возвращает через
543. /// модификатор out число, полученное в результате
544. /// парсинга. </param>
545. /// <param name="min"> Минимальное значение,
546. /// возможное для параметра. </param>
547. /// <param name="max"> Максимальное значение,
548. /// возможное для параметра. </param>
549. /// <returns></returns>
550. public bool CheckValidityFailedI(string newText, out int number,
551. int min, int max)
552. {
553. number = 0;
554. if (string.IsNullOrEmpty(newText))
555. {
556. return true;
557. }
558. if (!int.TryParse(newText, out number))
559. {
560. MessageBox.Show("Вы не можете изменить значение " +
561. "ячейки на такое значение!");
562. return true;
563. }
564. if (number < min || number > max)
565. {
566. MessageBox.Show("Число находится вне " +
567. "границ, допустимых для этого параметра!");
568. return true;
569. }
570. return false;
571. }
572. /// <summary>
573. /// Обработчик покидания текстового поля для задания
574. /// параметра N. Вызывается метод изменения значения
575. /// параметра с проверкой корректности.
576. /// </summary>
577. /// <param name="sender"></param>
578. /// <param name="e"></param>
579. private void InN\_Leave(object sender, EventArgs e)
580. {
581. ChangeValueI(ref N, InN.Text, InN, lN, 1);
582. }
583. /// <summary>
584. /// Обработчик покидания текстового поля для задания
585. /// параметра Mv. Вызывается метод изменения значения
586. /// параметра с проверкой корректности.
587. /// </summary>
588. /// <param name="sender"></param>
589. /// <param name="e"></param>
590. private void InMv\_Leave(object sender, EventArgs e)
591. {
592. ChangeValueI(ref Mv, InMv.Text, InMv, lMv, 1);
593. }
594. /// <summary>
595. /// Обработчик покидания текстового поля для задания
596. /// параметра Mu. Вызывается метод изменения значения
597. /// параметра с проверкой корректности.
598. /// </summary>
599. /// <param name="sender"></param>
600. /// <param name="e"></param>
601. private void InMu\_Leave(object sender, EventArgs e)
602. {
603. ChangeValueI(ref Mu,InMu.Text, InMu, lMu, 1);
604. }
605. /// <summary>
606. /// Обработчик покидания текстового поля для задания
607. /// параметра mu. Вызывается метод изменения значения
608. /// параметра с проверкой корректности.
609. /// </summary>
610. /// <param name="sender"></param>
611. /// <param name="e"></param>
612. private void In\_mu\_Leave(object sender, EventArgs e)
613. {
614. ChangeValueF(ref \_mu, In\_mu.Text, In\_mu, l\_mu);
615. }
616. /// <summary>
617. /// Обработчик покидания текстового поля для задания
618. /// параметра Um. Вызывается метод изменения значения
619. /// параметра с проверкой корректности.
620. /// </summary>
621. /// <param name="sender"></param>
622. /// <param name="e"></param>
623. private void InUm\_Leave(object sender, EventArgs e)
624. {
625. ChangeValueF(ref Um, InUm.Text, InUm, lUm, 0 + float.Epsilon);
626. }
627. /// <summary>
628. /// Обработчик покидания текстового поля для задания
629. /// параметра Vm. Вызывается метод изменения значения
630. /// параметра с проверкой корректности.
631. /// </summary>
632. /// <param name="sender"></param>
633. /// <param name="e"></param>
634. private void InVm\_Leave(object sender, EventArgs e)
635. {
636. ChangeValueF(ref Vm, InVm.Text, InVm, lVm, 0 + float.Epsilon);
637. }
638. /// <summary>
639. /// Обработчик покидания текстового поля для задания
640. /// параметра du. Вызывается метод изменения значения
641. /// параметра с проверкой корректности.
642. /// </summary>
643. /// <param name="sender"></param>
644. /// <param name="e"></param>
645. private void Indu\_Leave(object sender, EventArgs e)
646. {
647. ChangeValueF(ref du,Indu.Text, Indu, ldu);
648. }
649. /// <summary>
650. /// Обработчик покидания текстового поля для задания
651. /// параметра dv. Вызывается метод изменения значения
652. /// параметра с проверкой корректности.
653. /// </summary>
654. /// <param name="sender"></param>
655. /// <param name="e"></param>
656. private void Indv\_Leave(object sender, EventArgs e)
657. {
658. ChangeValueF(ref dv, Indv.Text, Indv, ldv);
659. }
660. /// <summary>
661. /// Обработчик покидания текстового поля для задания
662. /// параметра u. Вызывается метод изменения значения
663. /// параметра с проверкой корректности.
664. /// </summary>
665. /// <param name="sender"></param>
666. /// <param name="e"></param>
667. private void Inu\_Leave(object sender, EventArgs e)
668. {
669. ChangeValueF(ref u,Inu.Text, Inu, lu, (float)(-1 / 12.0), (float)(1 / 12.0));
670. }
671. /// <summary>
672. /// Обработчик покидания текстового поля для задания
673. /// параметра v. Вызывается метод изменения значения
674. /// параметра с проверкой корректности.
675. /// </summary>
676. /// <param name="sender"></param>
677. /// <param name="e"></param>
678. private void Inv\_Leave(object sender, EventArgs e)
679. {
680. ChangeValueF(ref v, Inv.Text,Inv, lv, (float)(-1 / 12.0), (float)(1 / 12.0));
681. }
682. /// <summary>
683. /// Обработчик покидания текстового поля для задания
684. /// параметра a. Вызывается метод изменения значения
685. /// параметра с проверкой корректности.
686. /// </summary>
687. /// <param name="sender"></param>
688. /// <param name="e"></param>
689. private void Ina\_Leave(object sender, EventArgs e)
690. {
691. ChangeValueF(ref a, Ina.Text,Ina, la);
692. }
693. /// <summary>
694. /// Обработчик покидания текстового поля для задания
695. /// параметра Tc. Вызывается метод изменения значения
696. /// параметра с проверкой корректности.
697. /// </summary>
698. /// <param name="sender"></param>
699. /// <param name="e"></param>
700. private void InTc\_Leave(object sender, EventArgs e)
701. {
702. ChangeValueF(ref Tc, InTc.Text, InTc, lTc);
703. }
704. /// <summary>
705. /// Обработчик покидания текстового поля для задания
706. /// параметра H. Вызывается метод изменения значения
707. /// параметра с проверкой корректности.
708. /// </summary>
709. /// <param name="sender"></param>
710. /// <param name="e"></param>
711. private void InH\_Leave(object sender, EventArgs e)
712. {
713. ChangeValueF(ref H, InH.Text, InH, lH, 0 + float.Epsilon);
714. }
715. /// <summary>
716. /// Обработчик покидания текстового поля для задания
717. /// параметра up1. Вызывается метод изменения значения
718. /// параметра с проверкой корректности.
719. /// </summary>
720. /// <param name="sender"></param>
721. /// <param name="e"></param>
722. private void Inup1\_Leave(object sender, EventArgs e)
723. {
724. ChangeValueF(ref up1, Inup1.Text, Inup1, lup1);
725. }
726. /// <summary>
727. /// Обработчик покидания текстового поля для задания
728. /// параметра up2. Вызывается метод изменения значения
729. /// параметра с проверкой корректности.
730. /// </summary>
731. /// <param name="sender"></param>
732. /// <param name="e"></param>
733. private void Inup2\_Leave(object sender, EventArgs e)
734. {
735. ChangeValueF(ref up2, Inup2.Text, Inup2, lup2);
736. }
737. /// <summary>
738. /// Обработчик покидания текстового поля для задания
739. /// параметра vp1. Вызывается метод изменения значения
740. /// параметра с проверкой корректности.
741. /// </summary>
742. /// <param name="sender"></param>
743. /// <param name="e"></param>
744. private void Invp1\_Leave(object sender, EventArgs e)
745. {
746. ChangeValueF(ref vp1, Invp1.Text, Invp1, lvp1);
747. }
748. /// <summary>
749. /// Обработчик покидания текстового поля для задания
750. /// параметра vp2. Вызывается метод изменения значения
751. /// параметра с проверкой корректности.
752. /// </summary>
753. /// <param name="sender"></param>
754. /// <param name="e"></param>
755. private void Invp2\_Leave(object sender, EventArgs e)
756. {
757. ChangeValueF(ref vp2, Invp2.Text, Invp2, lvp2);
758. }
759. /// <summary>
760. /// Обработчик покидания текстового поля для задания
761. /// параметра Ap1. Вызывается метод изменения значения
762. /// параметра с проверкой корректности.
763. /// </summary>
764. /// <param name="sender"></param>
765. /// <param name="e"></param>
766. private void InAp1\_Leave(object sender, EventArgs e)
767. {
768. ChangeValueF(ref Ap1, InAp1.Text, InAp1, lAp1);
769. }
770. /// <summary>
771. /// Обработчик покидания текстового поля для задания
772. /// параметра Ap2. Вызывается метод изменения значения
773. /// параметра с проверкой корректности.
774. /// </summary>
775. /// <param name="sender"></param>
776. /// <param name="e"></param>
777. private void InAp2\_Leave(object sender, EventArgs e)
778. {
779. ChangeValueF(ref Ap2, InAp2.Text, InAp2, lAp2);
780. }
781. /// <summary>
782. /// Обработчик покидания текстового поля для задания
783. /// параметра gamma. Вызывается метод изменения значения
784. /// параметра с проверкой корректности.
785. /// </summary>
786. /// <param name="sender"></param>
787. /// <param name="e"></param>
788. private void Ingamma\_Leave(object sender, EventArgs e)
789. {
790. ChangeValueF(ref gamma, Ingamma.Text, Ingamma, lgamma);
791. }
792. /// <summary>
793. /// Обработчик покидания текстового поля для задания
794. /// параметра pq. Вызывается метод изменения значения
795. /// параметра с проверкой корректности.
796. /// </summary>
797. /// <param name="sender"></param>
798. /// <param name="e"></param>
799. private void Inpq\_Leave(object sender, EventArgs e)
800. {
801. ChangeValueI(ref pq, Inpq.Text, Inpq, lpq, 1);
802. }
803. /// <summary>
804. /// Обработчик нажатия на кнопку сохранения параметров.
805. /// Появляется окно выбора директории, затем файл
806. /// записывается в выбранное место. Также обрабатываются
807. /// исключения, которые могут возникнуть при записи.
808. /// </summary>
809. /// <param name="sender"></param>
810. /// <param name="e"></param>
811. private void bSave\_Click(object sender, EventArgs e)
812. {
813. string path = "";
814. if (saveFileInput.ShowDialog()
815. == DialogResult.OK)
816. {
817. path = saveFileInput.FileName;
818. }
819. try
820. {
821. File.WriteAllLines(path, new string[]
822. { N.ToString(), Mu.ToString(), Mv.ToString(),Um.ToString(), Vm.ToString(),
823. du.ToString(), dv.ToString(), u.ToString(), v.ToString(), a.ToString(),
824. Tc.ToString(), up1.ToString(), up2.ToString(), vp1.ToString(),vp2.ToString(),
825. Ap1.ToString(),Ap2.ToString(), gamma.ToString(), \_mu.ToString(), H.ToString(),
826. pq.ToString() });
827. }
828. catch (FileNotFoundException)
829. {
830. MessageBox.Show("Файл не существует. " +
831. "Начните заново.");
832. }
833. catch (IOException)
834. {
835. MessageBox.Show("Ошибка в работе с файлом." +
836. " Начните заново.");
837. }
838. catch (UnauthorizedAccessException)
839. {
840. MessageBox.Show("Ошибка доступа к файлу:" +
841. " нет разрешения на доступ. Начните заново.");
842. }
843. catch (System.Security.SecurityException)
844. {
845. MessageBox.Show("Ошибка безопасности при " +
846. "работе с файлом. Начните заново.");
847. }
848. catch (Exception ex)
849. {
850. MessageBox.Show("Произошла ошибка: " + ex.Message);
851. }
852. finally { MessageBox.Show("Работа с файлом окончена."); }
853. }
854. /// <summary>
855. /// Обработчик нажатия на кнопку перехода к
856. /// дальнейшей работе с данными, происходит
857. /// запись файла входных параметров, создание
858. /// и открытие новой формы на основе текущей,
859. /// сокрытие текущей формы.
860. /// </summary>
861. /// <param name="sender"></param>
862. /// <param name="e"></param>
863. private void bProceed\_Click(object sender, EventArgs e)
864. {
865. try
866. {
867. File.WriteAllLines("Input.txt", new string[]
868. { N.ToString(), Mu.ToString(), Mv.ToString(),Um.ToString(), Vm.ToString(),
869. du.ToString(), dv.ToString(), u.ToString(), v.ToString(), a.ToString(),
870. Tc.ToString(), up1.ToString(), up2.ToString(), vp1.ToString(),vp2.ToString(),
871. Ap1.ToString(),Ap2.ToString(), gamma.ToString(), \_mu.ToString(), H.ToString(),
872. pq.ToString() });
873. }
874. catch (Exception ex)
875. {
876. MessageBox.Show("Произошла ошибка: " + ex.Message);
877. }
878. this.Hide();
879. new Operations(this, prevForm, funcs, N, Mu, Mv, Um, Vm, du, dv, u, v, a, Tc,
880. up1, up2, vp1, vp2, Ap1, Ap2, gamma, \_mu, H, pq).Show();
881. }
882. /// <summary>
883. /// Обработчик кнопки закрытия формы.
884. /// </summary>
885. /// <param name="sender"></param>
886. /// <param name="e"></param>
887. private void bExit\_Click(object sender, EventArgs e)
888. {
889. this.Close();
890. }
891. /// <summary>
892. /// Обработчик закрытия формы, который
893. /// открывает предыдущую форму
894. /// </summary>
895. /// <param name="sender"></param>
896. /// <param name="e"></param>
897. private void Input\_FormClosing(object sender,
898. FormClosingEventArgs e)
899. {
900. prevForm.Show();
901. }
902. /// <summary>
903. /// Метод, позволяющий получить все элементы
904. /// определенного типа на форме для их
905. /// более удобной настройки.
906. /// </summary>
907. /// <typeparam name="T"> Параметр для типа
908. /// желаемых элементов. </typeparam>
909. /// <param name="container"> Параметр для
910. /// контейнера, на котором требуется найти
911. /// элементы желаемого типа. </param>
912. /// <returns> Возвращает список элементов
913. /// контроля желаемого типа. </returns>
914. static private List<Control> GetAllControlsOfType<T>(Control container)
915. {
916. List<Control> controlList = new List<Control>();
917. foreach (Control c in container.Controls)
918. {
919. controlList.AddRange(GetAllControlsOfType<T>(c));
920. if (c is T)
921. {
922. controlList.Add(c);
923. }
924. }
925. return controlList;
926. }
927. }
928. }
     * 1. Код класса Operations.cs
929. using System;
930. using System.Collections.Generic;
931. using System.Drawing;
932. using System.Linq;
933. using System.Text;
934. using System.Threading.Tasks;
935. using System.Windows.Forms;
936. using RadarLib;
937. using System.IO;
938. using MatlabFuncs;
939. using MathWorks.MATLAB.NET.Arrays;
940. namespace Coursework
941. {
942. /// <summary>
943. /// Форма для загрузки, генерирования и сохранения
944. /// файлов наблюдений, корреляционной матрицы,
945. /// статистики наблюдений и визуализации.
946. /// </summary>
947. public partial class Operations : Form
948. {
949. /// <summary>
950. /// Поля для цветов, которые определяют внешний
951. /// вид формы.
952. /// </summary>
953. Color backColor = Color.FromArgb(46, 76, 86);
954. Color panelColor = Color.FromArgb(89, 139, 155);
955. Color hoverColor = Color.FromArgb(125, 167, 181);
956. Color textColor = Color.FromArgb(160, 238, 230);
957. /// <summary>
958. /// Поля для предыдущей формы и начального экрана.
959. /// </summary>
960. Input prevForm;
961. Homescreen homescreen;
962. /// <summary>
963. /// Поле для экземпляра класса функций матлаб.
964. /// </summary>
965. Funcs funcs;
966. /// <summary>
967. /// Поля для параметров симуляции.
968. /// </summary>
969. float Um, Vm, du, dv, u, v, a, Tc, up1,
970. up2, vp1, vp2, Ap1, Ap2, gamma, \_mu, H;
971. int N, Mu, Mv, pq, M;
972. /// <summary>
973. /// Поля-флаги для определения стадии работы
974. /// и мониторинга добавленной информации и
975. /// опций, открытых в зависимости от этого.
976. /// </summary>
977. bool dataE, matrixE, statE;
978. /// <summary>
979. /// Конструктор для формы с параметрами - параметрами симуляции.
980. /// Задает внешний вид формы, значения полей, настраивает
981. /// подсказки для пользователя.
982. /// </summary>
983. /// <param name="form"> Параметр для предыдущей формы. </param>
984. /// <param name="homescreen"> Параметр для начального экрана. </param>
985. /// <param name="funcs"> Параметр для экземпляра класса функций
986. /// матлаб. </param>
987. /// <param name="N"> Параметр для количества векторов, по которым
988. /// составляются наблюдения. </param>
989. /// <param name="Mu"> Параметр для размера решетки радиолокатора
990. /// по длине. </param>
991. /// <param name="Mv"> Параметр для размера решетки радиолокатора
992. /// по ширине. </param>
993. /// <param name="Um"> Параметр для ширины главного лепестка
994. /// диаграммы направленности модуля по одному углу. </param>
995. /// <param name="Vm"> Параметр для ширины главного лепестка
996. /// диаграммы направленности модуля по другому углу. </param>
997. /// <param name="du"> Параметр для сдвига до центра
998. /// просматриваемого сектора по одному углу. </param>
999. /// <param name="dv"> Параметр для сдвига до центра
1000. /// просматриваемого сектора по другому углу. </param>
1001. /// <param name="u"> Параметр для предполагаемого положения
1002. /// цели по одному углу. </param>
1003. /// <param name="v"> Параметр для предполагаемого положения
1004. /// цели по одному углу. </param>
1005. /// <param name="a"> Параметр для мощности сигнала. </param>
1006. /// <param name="Tc"> Параметр для времени начала прихода
1007. /// сигнала. </param>
1008. /// <param name="up1"> Параметр для первой компоненты
1009. /// положения помех по одному углу. </param>
1010. /// <param name="up2"> Параметр для второй компоненты
1011. /// положения помех по одному углу. </param>
1012. /// <param name="vp1"> Параметр для первой компоненты
1013. /// положения помех по другому углу. </param>
1014. /// <param name="vp2"> Параметр для второй компоненты
1015. /// положения помех по другому углу. </param>
1016. /// <param name="Ap1"> Параметр для первой компоненты
1017. /// мощности помех. </param>
1018. /// <param name="Ap2"> Параметр для второй компоненты
1019. /// мощности помех. </param>
1020. /// <param name="gamma"> Параметр гамма. </param>
1021. /// <param name="\_mu"> Параметр для коэффициента
1022. /// регуляризации корреляционной матрицы. </param>
1023. /// <param name="H"> Параметр для порогового значения
1024. /// мощности сигнала. </param>
1025. /// <param name="pq"> Параметр для количества блоков,
1026. /// по которым составляется корреляционная матрица. </param>
1027. public Operations(Input form, Homescreen homescreen,
1028. Funcs funcs, int N, int Mu, int Mv,
1029. float Um, float Vm, float du, float dv, float u, float v,
1030. float a, float Tc, float up1, float up2, float vp1, float vp2,
1031. float Ap1, float Ap2, float gamma, float \_mu, float H, int pq)
1032. {
1033. this.Mu = Mu;
1034. this.Mv = Mv;
1035. this.Um = Um;
1036. this.Vm = Vm;
1037. this.u = u;
1038. this.v = v;
1039. this.du = du;
1040. this.dv = dv;
1041. this.a = a;
1042. this.Tc = Tc;
1043. this.up1 = up1;
1044. this.up2 = up2;
1045. this.vp1 = vp1;
1046. this.vp2 = vp2;
1047. this.Ap1 = Ap1;
1048. this.Ap2 = Ap2;
1049. this.H = H;
1050. this.\_mu = \_mu;
1051. this.pq = pq;
1052. this.N = N;
1053. this.gamma = gamma;
1054. this.funcs = funcs;
1055. prevForm = form;
1056. this.homescreen = homescreen;
1057. M = Mu \* Mv;
1058. InitializeComponent();
1059. log.Text = "Параметры симуляции определены";
1060. dataE = matrixE = statE = bSaveData.Enabled
1061. = bGenMatrix.Enabled = bSaveMatrix.Enabled
1062. = bGenStat.Enabled = bSaveStat.Enabled
1063. = bVisualize.Enabled = false;
1064. this.BackColor = log.BackColor = backColor;
1065. panel.BackColor = panelColor;
1066. log.ForeColor = textColor;
1067. foreach (Button b in GetAllControlsOfType<Button>(this))
1068. {
1069. b.FlatAppearance.MouseOverBackColor = hoverColor;
1070. b.ForeColor = textColor;
1071. }
1072. toolTip.SetToolTip(bBackToHomepage, "Внимание!" +
1073. "\nПри возвращении прогресс текущей сессии будет потерян." +
1074. "\nУбедитесь в том, что вы сохранили все файлы," +
1075. "\nкоторые не хотите потерять.");
1076. toolTip.SetToolTip(bBackToInput, "Внимание!" +
1077. "\nПри возвращении прогресс текущей сессии," +
1078. "\nкроме параметров симуляции, будет потерян." +
1079. "\nУбедитесь в том, что вы сохранили все файлы," +
1080. "\nкоторые не хотите потерять.");
1081. }
1082. /// <summary>
1083. /// Обработчик нажатия на кнопку добавления файлов наблюдений.
1084. /// Вызывает метод открытия желаемых файлов с проверкой длины,
1085. /// обновляет информацию в логгере, в случае успеха меняет
1086. /// флаг наличия файлов наблюдений и активизирует связанные с
1087. /// этим кнопки.
1088. /// </summary>
1089. /// <param name="sender"></param>
1090. /// <param name="e"></param>
1091. private void bGetData\_Click(object sender, EventArgs e)
1092. {
1093. log.Text += "\nДобавляем файлы наблюдений";
1094. progress.Visible = true;
1095. if (OpenFile("Sx.txt", false, 0) && OpenFile("Sy.txt", false, 0)
1096. && OpenFile("Ux.txt", false, 0) && OpenFile("Uy.txt", false, 0)
1097. && OpenFile("YX.txt", true, M \* N \* pq)
1098. && OpenFile("YY.txt", true, M \* N \* pq))
1099. {
1100. progress.Visible = false;
1101. log.Text += "\nФайлы наблюдений добавлены";
1102. dataE = bGenMatrix.Enabled = bSaveData.Enabled = true;
1103. }
1104. else
1105. {
1106. progress.Visible = false;
1107. log.Text += "\nПроизошла ошибка добавления файлов.";
1108. }
1109. }
1110. /// <summary>
1111. /// Обработчик нажатия на кнопку добавления файлов матрицы.
1112. /// Вызывает метод открытия желаемых файлов с проверкой длины,
1113. /// обновляет информацию в логгере, в случае успеха меняет
1114. /// флаг наличия файлов матрицы и активизирует связанные с
1115. /// этим кнопки.
1116. /// </summary>
1117. /// <param name="sender"></param>
1118. /// <param name="e"></param>
1119. private void bGetMatrix\_Click(object sender, EventArgs e)
1120. {
1121. log.Text += "\nДобавляем файлы матрицы";
1122. progress.Visible = true;
1123. if (OpenFile("Rx.txt", true, M \* M \* pq)
1124. && OpenFile("Ry.txt", true, M \* M \* pq))
1125. {
1126. progress.Visible = false;
1127. log.Text += "\nФайлы матрицы добавлены";
1128. matrixE = bGenStat.Enabled = bSaveMatrix.Enabled = true;
1129. }
1130. else
1131. {
1132. progress.Visible = false;
1133. log.Text += "\nПроизошла ошибка добавления файлов.";
1134. }
1135. }
1136. /// <summary>
1137. /// Обработчик нажатия на кнопку добавления файла статистики.
1138. /// Вызывает метод открытия желаемого файла с проверкой длины,
1139. /// обновляет информацию в логгере, в случае успеха меняет
1140. /// флаг наличия файла статистики и активизирует связанные с
1141. /// этим кнопки.
1142. /// </summary>
1143. /// <param name="sender"></param>
1144. /// <param name="e"></param>
1145. private void bGetStat\_Click(object sender, EventArgs e)
1146. {
1147. log.Text += "\nДобавляем файл статистики";
1148. progress.Visible = true;
1149. if (OpenFile("Lfile.txt", true, 9 \* 9 \* N / 9))
1150. {
1151. progress.Visible = false;
1152. log.Text += "\nФайл статистики добавлен";
1153. statE = bVisualize.Enabled = bSaveStat.Enabled = true;
1154. }
1155. else
1156. {
1157. progress.Visible = false;
1158. log.Text += "\nПроизошла ошибка добавления файлов.";
1159. }
1160. }
1161. /// <summary>
1162. /// Метод для добавления информации из файлов.
1163. /// Открывает диалоговое окно для выбора файла,
1164. /// вызывает метод проверки корректности файла,
1165. /// в случае корректности данных копирует
1166. /// выбранный файл в соответсвующий рабочий файл,
1167. /// сохраняет изменения в логгер, обрабатывает
1168. /// исключения, которые могут возникнуть при
1169. /// открытии и чтении файла.
1170. /// </summary>
1171. /// <param name="name"> Параметр для имени
1172. /// рабочего файла, в который должна быть
1173. /// скопирована информация. </param>
1174. /// <param name="checkLen"> Параметр для флага,
1175. /// нужно ли проверять длину файла. </param>
1176. /// <param name="len"> Параметр для ожидаемой
1177. /// длины файла. </param>
1178. /// <returns> возвращает результат успешности
1179. /// добавления файла. </returns>
1180. bool OpenFile(string name, bool checkLen, int len)
1181. {
1182. try
1183. {
1184. log.Text += "\nДобавляем файл " + name;
1185. MessageBox.Show("Вам нужно выбрать файл " + name);
1186. string path;
1187. openFile.FileName = null;
1188. openFile.ShowDialog();
1189. path = openFile.FileName;
1190. if (CheckData(path, checkLen, len))
1191. {
1192. File.Copy(path, name, true);
1193. log.Text += "\nФайл " + name + " добавлен";
1194. return true;
1195. }
1196. else return false;
1197. }
1198. catch (ArgumentException ex)
1199. {
1200. MessageBox.Show("Вы не выбрали путь для файла.\n" + ex.Message);
1201. log.Text += "\nОшибка при добавлении файла " + name;
1202. return false;
1203. }
1204. catch (FileNotFoundException ex)
1205. {
1206. MessageBox.Show("Файл не существует. " +
1207. "Начните заново.\n" + ex.Message);
1208. log.Text += "\nОшибка при добавлении файла " + name;
1209. return false;
1210. }
1211. catch (IOException ex)
1212. {
1213. MessageBox.Show("Ошибка в работе с файлом." +
1214. " Начните заново.\n" + ex.Message);
1215. log.Text += "\nОшибка при добавлении файла " + name;
1216. return false;
1217. }
1218. catch (UnauthorizedAccessException ex)
1219. {
1220. MessageBox.Show("Ошибка доступа к файлу:" +
1221. " нет разрешения на доступ. Начните заново.\n" + ex.Message);
1222. log.Text += "\nОшибка при добавлении файла " + name;
1223. return false;
1224. }
1225. catch (System.Security.SecurityException ex)
1226. {
1227. MessageBox.Show("Ошибка безопасности при " +
1228. "работе с файлом. Начните заново.\n" + ex.Message);
1229. log.Text += "\nОшибка при добавлении файла " + name;
1230. return false;
1231. }
1232. catch (Exception ex)
1233. {
1234. MessageBox.Show("Произошла ошибка: " + ex.Message);
1235. log.Text += "\nОшибка при добавлении файла " + name;
1236. return false;
1237. }
1238. }
1239. /// <summary>
1240. /// Метод для проверки корректности загружаемых файлов.
1241. /// Вызывает метод считывания чисел и проверяет соответствие
1242. /// каждого числа требуемому типу, также сравнивает длину
1243. /// файла с ожидаемой длиной и возвращает результат соответствия
1244. /// файла требованиям.
1245. /// </summary>
1246. /// <param name="path"> Параметр для пути к проверяемому
1247. /// файлу. </param>
1248. /// <param name="checkLen"> Параметр для флага, нужна ли
1249. /// для этого файла проверка длины. </param>
1250. /// <param name="expectedLen"> Параметр для ожидаемой длины
1251. /// файла. </param>
1252. /// <returns> возвращает булевое true, если файл содержит
1253. /// числа в нужном количестве, и false в противном случае. </returns>
1254. bool CheckData(string path, bool checkLen, int expectedLen)
1255. {
1256. StreamReader filestr =
1257. new StreamReader(new FileStream(path, FileMode.Open));
1258. float num;
1259. int length = 0;
1260. while (true)
1261. {
1262. var t = NextNumber(filestr);
1263. if (t == "") break;
1264. if (!float.TryParse(t, out num))
1265. {
1266. MessageBox.Show($"В файле {path} находятся данные" +
1267. $" не в требуемом числовом формате");
1268. return false;
1269. }
1270. length++;
1271. }
1272. filestr.Dispose();
1273. if (checkLen && (expectedLen > length))
1274. {
1275. MessageBox.Show($"В файле {path} находятся данные" +
1276. $" не в требуемом количестве {length}");
1277. return false;
1278. }
1279. return true;
1280. }
1281. /// <summary>
1282. /// Метод для считывания следующего числа из
1283. /// файлового потока.
1284. /// </summary>
1285. /// <param name="streamReader"> Параметр для потока
1286. /// чтения. </param>
1287. /// <returns> Возвращает следующее число или пустую
1288. /// строку, если файл закончился. </returns>
1289. string NextNumber(StreamReader streamReader)
1290. {
1291. int c = streamReader.Peek();
1292. while (c != -1 && Char.IsWhiteSpace(Convert.ToChar(c)))
1293. {
1294. streamReader.Read();
1295. c = streamReader.Peek();
1296. }
1297. if (c == -1) return "";
1298. StringBuilder b = new StringBuilder();
1299. while (c != -1 && !Char.IsWhiteSpace(Convert.ToChar(c)))
1300. {
1301. b.Append(Convert.ToChar(c));
1302. streamReader.Read();
1303. c = streamReader.Peek();
1304. }
1305. return b.ToString();
1306. }
1307. /// <summary>
1308. /// Обработчик нажатия на кнопку генерации наблюдений.
1309. /// Вызывает асинхронный метод генерации наблюдений.
1310. /// </summary>
1311. /// <param name="sender"></param>
1312. /// <param name="e"></param>
1313. private void bGenData\_Click(object sender, EventArgs e)
1314. {
1315. BeginData();
1316. }
1317. /// <summary>
1318. /// Метод для генерирования наблюдений, настраивает
1319. /// вид формы на время осуществления процесса и вызывает
1320. /// метод генерирования наблюдений из экемпляра класса
1321. /// функций матлаб, засекает время за которое процесс
1322. /// завершается, обновляет информацию в логгере и флаг
1323. /// наличия наблюдений в случае успеха.
1324. /// </summary>
1325. async void BeginData()
1326. {
1327. try
1328. {
1329. Disable();
1330. progress.Visible = true;
1331. log.Text += "\nНачинаем генерацию наблюдений";
1332. MWArray[] res = null;
1333. await Task.Run(() =>
1334. {
1335. res = funcs.CreateFiles(1, N, Mu, Mv, Um, Vm, du, dv,
1336. u, v, a, Tc, up1, up2, vp1, vp2, Ap1, Ap2, gamma, \_mu);
1337. });
1338. TimeSpan timeSpan = TimeSpan.FromSeconds(double.Parse(res[0].ToString()));
1339. string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
1340. timeSpan.Hours, timeSpan.Minutes, timeSpan.Seconds,
1341. timeSpan.Milliseconds / 10);
1342. progress.Visible = false;
1343. log.Text += $"\nНаблюдения сгенерированы за " + elapsedTime;
1344. dataE = true;
1345. }
1346. catch (Exception e)
1347. {
1348. MessageBox.Show("Произошла ошибка при генерации наблюдений." +
1349. " Начните заново, либо перейдите к другому шагу.\n" + e.Message);
1350. progress.Visible = false;
1351. log.Text += "\nПроизошла ошибка при генерации наблюдений";
1352. }
1353. finally { Enable(); }
1354. }
1355. /// <summary>
1356. /// Обработчик нажатия на кнопку расчета корреляционной матрицы.
1357. /// Вызывает асинхронный метод расчета корреляционной матрицы.
1358. /// </summary>
1359. /// <param name="sender"></param>
1360. /// <param name="e"></param>
1361. private void bGenMatrix\_Click(object sender, EventArgs e)
1362. {
1363. BeginMatrix();
1364. }
1365. /// <summary>
1366. /// Метод для расчета корреляционной матрицы, настраивает
1367. /// вид формы на время осуществления процесса и вызывает
1368. /// метод расчета матрицы из класса библиотеки, засекает
1369. /// время за которое процесс завершается, обновляет
1370. /// информацию в логгере и флаг наличия матрицы в случае успеха.
1371. /// </summary>
1372. async void BeginMatrix()
1373. {
1374. try
1375. {
1376. Disable();
1377. progress.Visible = true;
1378. log.Text += "\nНачинаем вычисление корреляционной матрицы";
1379. string res = "";
1380. await Task.Run(() => { res = RadarOperations.Start(Mu \* Mv, N, pq, \_mu); });
1381. progress.Visible = false;
1382. log.Text += $"\nВычисление матрицы закончено за {res}";
1383. matrixE = true;
1384. }
1385. catch (Exception e)
1386. {
1387. MessageBox.Show("Произошла ошибка при расчете корреляционной матрицы." +
1388. " Начните заново, либо перейдите к другому шагу.\n" + e.Message);
1389. progress.Visible = false;
1390. log.Text += "\nПроизошла ошибка при расчете корреляционной матрицы";
1391. }
1392. finally { Enable(); }
1393. }
1394. /// <summary>
1395. /// Обработчик нажатия на кнопку расчета статистики.
1396. /// Вызывает асинхронный метод расчета статистики.
1397. /// </summary>
1398. /// <param name="sender"></param>
1399. /// <param name="e"></param>
1400. private void bGenStat\_Click(object sender, EventArgs e)
1401. {
1402. BeginL();
1403. }
1404. /// <summary>
1405. /// Метод для расчета статистики наблюдений, настраивает
1406. /// вид формы на время осуществления процесса и вызывает
1407. /// метод расчета статистики наблюдений из экемпляра класса
1408. /// функций матлаб, засекает время за которое процесс
1409. /// завершается, обновляет информацию в логгере и флаг
1410. /// наличия статистики наблюдений в случае успеха.
1411. /// </summary>
1412. async void BeginL()
1413. {
1414. try
1415. {
1416. Disable();
1417. progress.Visible = true;
1418. log.Text += "\nНачинаем расчет статистики наблюдений";
1419. MWArray[] res = null;
1420. await Task.Run(() =>
1421. {
1422. res = funcs.CreateL(1, N, Mu, Mv, Um, Vm, \_mu);
1423. });
1424. TimeSpan timeSpan = TimeSpan.FromSeconds(double.Parse(res[0].ToString()));
1425. string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
1426. timeSpan.Hours, timeSpan.Minutes, timeSpan.Seconds,
1427. timeSpan.Milliseconds / 10);
1428. progress.Visible = false;
1429. log.Text += $"\nРасчет статистики наблюдений закончен за " + elapsedTime;
1430. statE = true;
1431. }
1432. catch (Exception e)
1433. {
1434. MessageBox.Show("Произошла ошибка при расчете статистики наблюдений." +
1435. " Начните заново, либо перейдите к другому шагу.\n" + e.Message);
1436. progress.Visible = false;
1437. log.Text += "\nПроизошла ошибка при расчете статистики наблюдений";
1438. }
1439. finally { Enable(); }
1440. }
1441. /// <summary>
1442. /// Обработчик нажатия на кнопку визуализации данных.
1443. /// Вызывает асинхронный метод визуализации.
1444. /// </summary>
1445. /// <param name="sender"></param>
1446. /// <param name="e"></param>
1447. private void bVisualize\_Click(object sender, EventArgs e)
1448. {
1449. BeginVis();
1450. }
1451. /// <summary>
1452. /// Метод для визуализации данных статистики, настраивает
1453. /// вид формы на время осуществления процесса и вызывает
1454. /// метод визуализации из экемпляра класса
1455. /// функций матлаб, засекает время за которое процесс
1456. /// завершается, обновляет информацию в логгере.
1457. /// </summary>
1458. async void BeginVis()
1459. {
1460. try
1461. {
1462. Disable();
1463. progress.Visible = true;
1464. log.Text += "\nНачинаем визуализацию";
1465. MWArray[] res = null;
1466. await Task.Run(() =>
1467. {
1468. res = funcs.CreatePicture(1, N, Um, Vm, H);
1469. });
1470. TimeSpan timeSpan = TimeSpan.FromSeconds(double.Parse(res[0].ToString()));
1471. string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
1472. timeSpan.Hours, timeSpan.Minutes, timeSpan.Seconds,
1473. timeSpan.Milliseconds / 10);
1474. progress.Visible = false;
1475. log.Text += $"\nВизуализация закончена за " + elapsedTime;
1476. }
1477. catch (Exception e)
1478. {
1479. MessageBox.Show("Произошла ошибка при визуализации." +
1480. " Начните заново, либо перейдите к другому шагу.\n" + e.Message);
1481. progress.Visible = false;
1482. log.Text += "\nПроизошла ошибка при визуализации";
1483. }
1484. finally { Enable(); }
1485. }
1486. /// <summary>
1487. /// Обработчик нажатия на кнопку сохранения файлов
1488. /// наблюдений. В логгер заносится запись о текущем
1489. /// процессе и вызывается метод копирования файла.
1490. /// </summary>
1491. /// <param name="sender"></param>
1492. /// <param name="e"></param>
1493. private void bSaveData\_Click(object sender, EventArgs e)
1494. {
1495. log.Text += "\nНачинаем сохранение файлов наблюдений";
1496. progress.Visible = true;
1497. SaveFile("Sx.txt");
1498. SaveFile("Sy.txt");
1499. SaveFile("Ux.txt");
1500. SaveFile("Uy.txt");
1501. SaveFile("YX.txt");
1502. SaveFile("YY.txt");
1503. progress.Visible = false;
1504. log.Text += "\nСохранение файлов наблюдений закончено";
1505. }
1506. /// <summary>
1507. /// Обработчик нажатия на кнопку сохранения файлов
1508. /// скорреляционной матрицы. В логгер заносится запись
1509. /// о текущем процессе и вызывается метод копирования файла.
1510. /// </summary>
1511. /// <param name="sender"></param>
1512. /// <param name="e"></param>
1513. private void bSaveMatrix\_Click(object sender, EventArgs e)
1514. {
1515. log.Text += "\nНачинаем сохранение файлов матрицы";
1516. progress.Visible = true;
1517. SaveFile("Rx.txt");
1518. SaveFile("Ry.txt");
1519. progress.Visible = false;
1520. log.Text += "\nСохранение файлов матрицы закончено";
1521. }
1522. /// <summary>
1523. /// Обработчик нажатия на кнопку сохранения файла
1524. /// статистики. В логгер заносится запись о текущем
1525. /// процессе и вызывается метод копирования файла.
1526. /// </summary>
1527. /// <param name="sender"></param>
1528. /// <param name="e"></param>
1529. private void bSaveStat\_Click(object sender, EventArgs e)
1530. {
1531. log.Text += "\nНачинаем сохранение файла статистики";
1532. progress.Visible = true;
1533. SaveFile("Lfile.txt");
1534. progress.Visible = false;
1535. log.Text += "\nСохранение файлов статистики закончено";
1536. }
1537. /// <summary>
1538. /// Метод для сохранения файлов в выбранное пользователем место.
1539. /// Заносит состояние задания в логгер, открывает диалоговое окно
1540. /// для сохранения файла и копирует рабочий файл в требуемую
1541. /// директорию. Также обрабатываютяс возможные исключения.
1542. /// </summary>
1543. /// <param name="name"> Параметр для имени копируемого файла.
1544. /// Он же адрес в силу организации работы с файлами в программе. </param>
1545. void SaveFile(string name)
1546. {
1547. try
1548. {
1549. log.Text += "\nСохраняем файл " + name;
1550. MessageBox.Show("Вам нужно выбрать, куда сохранить файл " + name);
1551. string path;
1552. saveFile.FileName = null;
1553. saveFile.ShowDialog();
1554. path = saveFile.FileName;
1555. File.Copy(name, path, true);
1556. log.Text += "\nФайл " + name + " сохранен";
1557. }
1558. catch (ArgumentException ex)
1559. {
1560. MessageBox.Show("Вы не выбрали путь для файла.\n" + ex.Message);
1561. log.Text += "\nОшибка при сохранении файла " + name;
1562. }
1563. catch (FileNotFoundException ex)
1564. {
1565. MessageBox.Show("Файл не существует. " +
1566. "Начните заново.\n" + ex.Message);
1567. log.Text += "\nОшибка при сохранении файла " + name;
1568. }
1569. catch (IOException ex)
1570. {
1571. MessageBox.Show("Ошибка в работе с файлом." +
1572. " Начните заново.\n" + ex.Message);
1573. log.Text += "\nОшибка при сохранении файла " + name;
1574. }
1575. catch (UnauthorizedAccessException ex)
1576. {
1577. MessageBox.Show("Ошибка доступа к файлу:" +
1578. " нет разрешения на доступ. Начните заново.\n" + ex.Message);
1579. log.Text += "\nОшибка при сохранении файла " + name;
1580. }
1581. catch (System.Security.SecurityException ex)
1582. {
1583. MessageBox.Show("Ошибка безопасности при " +
1584. "работе с файлом. Начните заново.\n" + ex.Message);
1585. log.Text += "\nОшибка при сохранении файла " + name;
1586. }
1587. catch (Exception ex)
1588. {
1589. MessageBox.Show("Произошла ошибка: " + ex.Message);
1590. log.Text += "\nОшибка при сохранении файла " + name;
1591. }
1592. }
1593. /// <summary>
1594. /// Обработчик нажатия на кнопку возвращения к
1595. /// предыдущей форме. Текущая форма закрывается
1596. /// и показывается предыдущая.
1597. /// </summary>
1598. /// <param name="sender"></param>
1599. /// <param name="e"></param>
1600. private void bBackToInput\_Click(object sender, EventArgs e)
1601. {
1602. this.Close();
1603. prevForm.Show();
1604. }
1605. /// <summary>
1606. /// Обработчик нажатия на кнопку возвращения на главную
1607. /// страницу. Закрываются предыдущая и текущая формы,
1608. /// открывается форма главной страницы.
1609. /// </summary>
1610. /// <param name="sender"></param>
1611. /// <param name="e"></param>
1612. private void bBackToHomepage\_Click(object sender, EventArgs e)
1613. {
1614. prevForm.Close();
1615. this.Close();
1616. homescreen.Show();
1617. }
1618. /// <summary>
1619. /// Метод, позволяющий получить все элементы
1620. /// определенного типа на форме для их
1621. /// более удобной настройки.
1622. /// </summary>
1623. /// <typeparam name="T"> Параметр для типа
1624. /// желаемых элементов. </typeparam>
1625. /// <param name="container"> Параметр для
1626. /// контейнера, на котором требуется найти
1627. /// элементы желаемого типа. </param>
1628. /// <returns> Возвращает список элементов
1629. /// контроля желаемого типа. </returns>
1630. static private List<Control> GetAllControlsOfType<T>(Control container)
1631. {
1632. List<Control> controlList = new List<Control>();
1633. foreach (Control c in container.Controls)
1634. {
1635. controlList.AddRange(GetAllControlsOfType<T>(c));
1636. if (c is T)
1637. {
1638. controlList.Add(c);
1639. }
1640. }
1641. return controlList;
1642. }
1643. /// <summary>
1644. /// Метод, который переводит кнопки в активное состояние
1645. /// в зависимости от того, могут ли они быть активными
1646. /// (была ли добавлена информация, с которой связана их
1647. /// работа).
1648. /// </summary>
1649. void Enable()
1650. {
1651. foreach (Button b in GetAllControlsOfType<Button>(this))
1652. {
1653. b.Enabled = true;
1654. }
1655. bVisualize.Enabled = bSaveStat.Enabled = statE;
1656. bGenStat.Enabled = bSaveMatrix.Enabled = matrixE;
1657. bGenMatrix.Enabled = bSaveData.Enabled = dataE;
1658. }
1659. /// <summary>
1660. /// Метод, переводящий все кнопки в неактивное состояние.
1661. /// </summary>
1662. void Disable()
1663. {
1664. foreach (Button b in GetAllControlsOfType<Button>(this))
1665. {
1666. b.Enabled = false;
1667. }
1668. }
1669. }
1670. }
      1. Код библиотеки RadarLib
         1. Код класса RadarOperations.cs
1671. using System;
1672. using System.Diagnostics;
1673. using System.IO;
1674. using System.Numerics;
1675. namespace RadarLib
1676. {
1677. /// <summary>
1678. /// Статический класс для расчета и записи в файлы обратной
1679. /// корреляционной матрицы наблюдений на основе параметров
1680. /// наблюдений и ранее сгенерированной матрицы наблюдений,
1681. /// считываемой из файлов.
1682. /// </summary>
1683. public static class RadarOperations
1684. {
1685. /// <summary>
1686. /// Вызываемый извне метод класса, который
1687. /// вызывает методы считывания матрицы наблюдений,
1688. /// расчета обратной корреляционной матрицы и
1689. /// записи ее в файл, кроме того осуществляется
1690. /// измерение времени всего процесса обработки
1691. /// информации, и оно возвращается в точку вызова.
1692. /// </summary>
1693. /// <param name="m"> Параметр для размеров приемной
1694. /// решетки радиолокатора. </param>
1695. /// <param name="n"> Параметр для количества векторов
1696. /// наблюдений, по которым составляется корреляционная
1697. /// матрица. </param>
1698. /// <param name="pq"> Параметр числа блоков, по которым
1699. /// составляются наблюдения и корреляционная матрица. </param>
1700. /// <param name="mu"> Параметр для коэффициента
1701. /// регуляризации корреляционной матрицы. </param>
1702. /// <param name="pathYx"> Параметр для пути к файлу, в
1703. /// котором находятся вещественные компоненты матрицы
1704. /// наблюдений. </param>
1705. /// <param name="pathYy"> Параметр для пути к файлу, в
1706. /// котором находятся мнимые компоненты матрицы
1707. /// наблюдений. </param>
1708. /// <param name="pathRx"> Параметр для пути к файлу, в
1709. /// котором находятся вещественные компоненты обратной
1710. /// корреляционной матрицы. </param>
1711. /// <param name="pathRy"> Параметр для пути к файлу, в
1712. /// котором находятся мнимые компоненты обратной
1713. /// корреляционной матрицы. </param>
1714. /// <returns> Метод возвращает строку с отформатированным
1715. /// временем работы по расчету и записи обратной
1716. /// корреляционной матрицы. </returns>
1717. public static string Start(int m, int n, int pq, float mu,
1718. string pathYx = "YX.txt", string pathYy = "YY.txt",
1719. string pathRx = "Rx.txt", string pathRy = "Ry.txt")
1720. {
1721. int dataSize = pq \* n \* m;
1722. int outSize = pq \* m \* m;
1723. Complex[] data, outMatrix;
1724. outMatrix = new Complex[outSize];
1725. data = new Complex[dataSize];
1726. Stopwatch stopwatch = new Stopwatch();
1727. stopwatch.Start();
1728. try
1729. {
1730. ReadData(m, n, pq, data, pathYx, pathYy);
1731. MatrixInv(m, n, pq, mu, data, outMatrix);
1732. WriteData(m, pq, outMatrix, pathRx, pathRy);
1733. }
1734. catch (Exception)
1735. {
1736. throw;
1737. }
1738. stopwatch.Stop();
1739. TimeSpan timeSpan = stopwatch.Elapsed;
1740. string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
1741. timeSpan.Hours, timeSpan.Minutes, timeSpan.Seconds,
1742. timeSpan.Milliseconds / 10);
1743. return elapsedTime;
1744. }
1745. /// <summary>
1746. /// Метод, с помощью которого через потоки считывается матрица
1747. /// наблюдений, сгенерированная ранее. Матрица, находящаяся
1748. /// в двух файлах (для вещественных и мнимых компонент),
1749. /// объединяется в массив комплексных чисел.
1750. /// </summary>
1751. /// <param name="m"> Параметр для размеров приемной
1752. /// решетки радиолокатора. </param>
1753. /// <param name="n"> Параметр для количества векторов
1754. /// наблюдений, по которым составляется корреляционная
1755. /// матрица. </param>
1756. /// <param name="pq"> Параметр числа блоков, по которым
1757. /// составляются наблюдения и корреляционная матрица. </param>
1758. /// <param name="mu"> Параметр для коэффициента
1759. /// регуляризации корреляционной матрицы. </param>
1760. /// <param name="data"> Ссылка на матрицу определенного
1761. /// размера, проинициализированную в основном методе класса,
1762. /// в которую будут записаны наблюдения, считанные из
1763. /// файлов. </param>
1764. /// <param name="pathYx"> Параметр для пути к файлу, в
1765. /// котором находятся вещественные компоненты матрицы
1766. /// наблюдений. </param>
1767. /// <param name="pathYy"> Параметр для пути к файлу, в
1768. /// котором находятся мнимые компоненты матрицы
1769. /// наблюдений. </param>
1770. static void ReadData(int m, int n, int pq, Complex[] data,
1771. string pathYx, string pathYy)
1772. {
1773. try
1774. {
1775. FileStream streamYx = new FileStream(pathYx, FileMode.Open);
1776. FileStream streamYy = new FileStream(pathYy, FileMode.Open);
1777. StreamReader readYx = new StreamReader(streamYx);
1778. StreamReader readYy = new StreamReader(streamYy);
1779. string[] linesYr = readYx.ReadToEnd().Split(new char[] { ' ' },
1780. StringSplitOptions.RemoveEmptyEntries);
1781. string[] linesYy = readYy.ReadToEnd().Split(new char[] { ' ' },
1782. StringSplitOptions.RemoveEmptyEntries);
1783. int it = 0;
1784. for (int j = 0; j < m; j++)
1785. {
1786. for (int k = 0; k < pq; k++)
1787. {
1788. for (int i = 0; i < n; i++)
1789. {
1790. data[pq \* m \* i + k \* m + j] =
1791. new Complex(float.Parse(linesYr[it].Trim(' ')),
1792. float.Parse(linesYy[it].Trim(' ')));
1793. it++;
1794. }
1795. }
1796. }
1797. readYx.Dispose();
1798. readYy.Dispose();
1799. }
1800. catch (Exception)
1801. {
1802. throw;
1803. }
1804. }
1805. /// <summary>
1806. /// Метод, осуществляющий расчет обратной корреляционной матрицы
1807. /// на основе матрицы наблюдений. В том числе происходит ее
1808. /// изначальное заполнение, затем расчет значений по матрице
1809. /// наблюдений и инверсия.
1810. /// </summary>
1811. /// /// <param name="m"> Параметр для размеров приемной
1812. /// решетки радиолокатора. </param>
1813. /// <param name="n"> Параметр для количества векторов
1814. /// наблюдений, по которым составляется корреляционная
1815. /// матрица. </param>
1816. /// <param name="pq"> Параметр числа блоков, по которым
1817. /// составляются наблюдения и корреляционная матрица. </param>
1818. /// <param name="mu"> Параметр для коэффициента
1819. /// регуляризации корреляционной матрицы. </param>
1820. /// <param name="data"> Ссылка на матрицу определенного
1821. /// размера, проинициализированную в основном методе класса,
1822. /// в которую будут записаны наблюдения, считанные из
1823. /// файлов. </param>
1824. /// <param name="outMatrix"> Ссылка на матрицу определенного
1825. /// размера, проинициализированную в основном методе класса,
1826. /// в которую будет записана рассчитанная корреляционная
1827. /// матрица. </param>
1828. static void MatrixInv(int m, int n, int pq, float mu,
1829. Complex[] data, Complex[] outMatrix)
1830. {
1831. try
1832. {
1833. float invMu = 1.0f / mu;
1834. float c;
1835. Complex[] a = new Complex[m];
1836. //Loop over all input matrices
1837. for (int k = 0; k < pq; k++)
1838. {
1839. //Intialize R0
1840. for (int j = 0; j < m; j++)
1841. {
1842. for (int o = 0; o < m; o++)
1843. {
1844. outMatrix[pq \* m \* j + k \* m + o] =
1845. new Complex((j == o) ? invMu : 0, 0);
1846. }
1847. }
1848. //N iterations for matrix inversion
1849. for (int i = 0; i < n; i++)
1850. {
1851. c = 1.0f;
1852. //Calculate a = R \* Y[i] matrix-vector product
1853. for (int j = 0; j < m; j++)
1854. {
1855. a[j] = new Complex(0, 0);
1856. for (int o = 0; o < m; o++)
1857. ComplexMAD(ref a[j], outMatrix[pq \* m \* j + k \* m + o],
1858. data[pq \* m \* i + k \* m + o]);
1859. //Isn't it more correct to accumulate C complex and
1860. //derive absolute value only in the end?
1861. c += (float)(a[j].Real \* data[pq \* m \* i + k \* m + j].Real
1862. + a[j].Imaginary \* data[pq \* m \* i + k \* m + j].Imaginary);
1863. }
1864. //Update R: R(n) = R(n - 1) - f(a)
1865. for (int j = 0; j < m; j++)
1866. {
1867. for (int o = 0; o < m; o++)
1868. {
1869. outMatrix[pq \* m \* j + k \* m + o] -=
1870. new Complex((a[j].Real \* a[o].Real +
1871. a[j].Imaginary \* a[o].Imaginary) / c,
1872. (-a[j].Real \* a[o].Imaginary +
1873. a[j].Imaginary \* a[o].Real) / c);
1874. }
1875. }
1876. }
1877. }
1878. }
1879. catch (Exception)
1880. {
1881. throw;
1882. }
1883. }
1884. /// <summary>
1885. /// Метод, выполняющий сложение первого передаваемого
1886. /// комплексного числа с произведением двух следующих -
1887. /// a += b \* c.
1888. /// </summary>
1889. /// <param name="a"> Параметр изменяемого числа, к
1890. /// которому прибавляется произведение двух следующих.</param>
1891. /// <param name="b"> Параметр для одного из чисел
1892. /// прибавляемого произведения. </param>
1893. /// <param name="c"> Параметр для другого из чисел
1894. /// прибавляемого произведения. </param>
1895. static void ComplexMAD(ref Complex a, Complex b, Complex c)
1896. {
1897. Complex t = new Complex(a.Real + b.Real \* c.Real - b.Imaginary \* c.Imaginary,
1898. a.Imaginary + b.Real \* c.Imaginary + b.Imaginary \* c.Real);
1899. a = t;
1900. }
1901. /// <summary>
1902. /// Метод, в котором через потоки в два файла
1903. /// (для вещественных и мнимых компонент) записывается
1904. /// рассчитанная корреляционная матрица.
1905. /// </summary>
1906. /// <param name="m"> Параметр для размеров приемной
1907. /// решетки радиолокатора. </param>
1908. /// <param name="pq"> Параметр числа блоков, по которым
1909. /// составляются наблюдения и корреляционная матрица. </param>
1910. /// <param name="outMatrix"> Ссылка на матрицу определенного
1911. /// размера, проинициализированную в основном методе класса,
1912. /// в которую будет записана рассчитанная корреляционная
1913. /// матрица. </param>
1914. /// <param name="pathRx"> Параметр для пути к файлу, в
1915. /// котором находятся вещественные компоненты обратной
1916. /// корреляционной матрицы. </param>
1917. /// <param name="pathRy"> Параметр для пути к файлу, в
1918. /// котором находятся мнимые компоненты обратной
1919. /// корреляционной матрицы. </param>
1920. static void WriteData(int m, int pq, Complex[] outMatrix,
1921. string pathRx, string pathRy)
1922. {
1923. try
1924. {
1925. FileStream streamRx = new FileStream(pathRx, FileMode.Create);
1926. FileStream streamRy = new FileStream(pathRy, FileMode.Create);
1927. StreamWriter writeRx = new StreamWriter(streamRx);
1928. StreamWriter writeRy = new StreamWriter(streamRy);
1929. for (int j = 0; j < m; j++)
1930. {
1931. for (int k = 0; k < pq; k++)
1932. {
1933. for (int i = 0; i < m; i++)
1934. {
1935. writeRx.Write(outMatrix[pq \* m \* j + k \* m + i].Real + " ");
1936. writeRy.Write(outMatrix[pq \* m \* j + k \* m + i].Imaginary + " ");
1937. }
1938. }
1939. writeRx.WriteLine();
1940. writeRy.WriteLine();
1941. }
1942. writeRx.Dispose();
1943. writeRy.Dispose();
1944. }
1945. catch (Exception)
1946. {
1947. throw;
1948. }
1949. }
1950. }
1951. }
      1. Код используемых MATLAB-функций
         1. Код функции CreateFiles.m
1952. function Toc=CreateFiles(N,Mu,Mv,Um,Vm,du,dv,u,v,a,Tc,up1,up2,vp1,vp2,Ap1,Ap2,gamma,mu)
1953. tic;
1955. M = Mu\*Mv; % ОБЩЕЕ ЧИСЛО МОДУЛЕЙ
1956. %----------------------------------------------------
1957. Up= Um/8; % ШИРИНА ДН ВСЕЙ РЕШЕТКИ ПО ОДНОЙ ОСИ
1958. Vp= Vm/8; % ШИРИНА ДН ВСЕЙ РЕШЕТКИ ПО ДРУГОЙ ОСИ
1959. %-----------------------------------------------------
1960. lu=6; % КОЛИЧЕСТВО СЕКТОРОВ ПО ОДНОЙ КООРДИНАТЕ
1961. lv=6; % КОЛИЧЕСТВО СЕКТОРОВ ПО ДРУГОЙ КООРДИНАТЕ
1962. lt=16; % ОПРЕДЕЛЯЕТСЯ ДАЛЬНОСТЬЮ ПРОСМОТРА
1963. %--------------------------------------------
1964. up=[up1 up2]; % ПОЛОЖЕНИЕ ПОМЕХ ПО ОДНОЙ КООРДИНАТЕ
1965. vp=[vp1 vp2]; % ПОЛОЖЕНИЕ ПОМЕХ ПО ДРУГОЙ КООРДИНАТЕ
1966. Ap=[Ap1 Ap2]; % МОЩНОСТЬ ПОМЕХ
1968. sig=0; % ТИП СИГНАЛА 0=импульс 1=ЛЧМ
1969. f=1; % ТИП ПРОСМОТРА ПО ВРЕМЕНИ, ТОЛЬКО ДЛЯ ИМПУЛЬСНОГО СИГНАЛА 1= через ls отсчетов 0=через 1 отсчет
1970. if sig
1971. %---------------------------------------------------
1972. ls=50;
1973. Kc=50;
1974. ll=0:.1:N-ls;
1975. Ns=length(ll);
1976. else
1977. Kc=0;
1978. ls=9;
1980. if f
1981. % ДЛИТЕЛЬНОСТЬ СИГНАЛА
1982. Ns=fix(N/ls); % КОЛИЧЕССТВО ПРОВЕРЯЕМЫХ ГИПОТЕЗ ПО ВРЕМЕНИ
1983. ll=1:ls:N-ls; % ЗНАЧЕНИЯ НАЧАЛА СИГНАЛА В ГИПОТЕЗАХ
1984. else
1985. Ns=N-ls;
1986. ll=1:N-ls;
1987. end
1988. end
1990. U\_=1\*Up; % ШАГ РАЗБИВКИ ПРОСТРАНСТВА В ДОЛЯХ ОТ ШИРИНЫ ДН РЕШЕТКИ
1991. ii=0:U\_:Um/2;
1992. ii=[-ii(end:-1:2) ii]; % ЗНАЧЕНИЯ ОДНОГО УГЛА В ГИПОТЕЗАХ
1993. Nu=length(ii); % КОЛИЧЕСТВО ПРОВЕРЯЕМЫХ ГИПОТЕЗ ПО ОДНОМУ УГЛУ
1994. %-----------------------------------------------------
1995. V\_=1\*Vp; % ШАГ РАЗБИВКИ ПРОСТРАНСТВА В ДОЛЯХ ОТ ШИРИНЫ ДН РЕШЕТКИ
1996. kk=0:V\_:Vm/2;
1997. kk=[-kk(end:-1:2) kk]; % ЗНАЧЕНИЯ ДРУГОГО УГЛА В ГИПОТЕЗАХ
1998. Nv=length(kk); % КОЛИЧЕСТВО ПРОВЕРЯЕМЫХ ГИПОТЕЗ ПО ДРУГОМУ УГЛУ
1999. %--ФОРМИРОВАНИЕ МАССИВА ВЕКТРОВ ОГИБАЮЩЕЙ СИГНАЛА ДЛЯ ВСЕХ ГИПОТЕЗ ПО УГЛУ-
2000. U=zeros(M,Nu\*Nv);
2002. [p,q]=ind2sub([Mu Mv],1:M);
2003. [i,k]=ind2sub([Nu Nv],1:Nu\*Nv);
2004. %U=exp(j\*2\*pi\*[p' q']\*[ii(i)/Um ; kk(k)/Vm]);
2006. g=sinx(pi\*ii(i)/Um).\*sinx(pi\*kk(k)/Vm);
2007. U=repmat(g,M,1).\*exp(j\*2\*pi\*[p' q']\*[ii(i)/Um ; kk(k)/Vm]);
2008. %---ФОРМИРОВАНИЕ МАССИВА ВЕКТОРОВ ДЛЯ ВСЕХ ГИПОТЕЗ ПО ВРЕМЕНИ---------
2009. S=zeros(N,Ns);
2010. L=zeros(Nu\*Nv,Ns);
2011. n=1:N;
2012. for l=1:Ns
2014. b=n-1>=ll(l)&n-1<ll(l)+ls;
2015. o=(n(b)-1-ll(l))/ls;
2016. S(n(b),l)=exp(-j\*pi\*Kc\*(o.^2-o));
2018. %S(:,l)=S(:,l)/sqrt(S(:,l)'\*S(:,l));
2019. S(:,l)=S(:,l)/sqrt(ls);
2021. end
2023. rU=real(U);
2024. iU=imag(U);
2026. rS=real(S);
2027. iS=imag(S);
2029. save Ux.txt rU -ascii;
2030. save Uy.txt iU -ascii;
2032. save Sx.txt rS -ascii;
2033. save Sy.txt iS -ascii;
2035. Y=zeros(Mu\*Mv,N,lu,lv,lt);
2036. R=zeros(Mu\*Mv,Mu\*Mv,lu,lv,lt);
2037. Rinv=zeros(Mu\*Mv,Mu\*Mv,lu,lv,lt);
2038. for i=1:lu
2039. for k=1:lv
2040. for l=1:lt
2041. Y(:,:,i,k,l)=getMatrNabl(N,Mu,Mv,Um,Vm,u-(i-1)\*du,v-(k-1)\*dv,a,Tc-(l-1)\*N,ls,Kc,up+(i-1)\*du,vp+(k-1)\*dv,Ap,gamma);
2042. C=Y(:,:,i,k,l);
2043. R(:,:,i,k,l)=Yinvert(C,N,Mu\*Mv,mu);
2044. end
2045. end
2046. end
2048. rY=real(Y(:,:));
2049. iY=imag(Y(:,:));
2051. save YX.txt rY -ascii;
2052. save YY.txt iY -ascii;
2053. Toc=toc;
2054. end
      * 1. Код функции CreateL.m
2055. function Toc=CreateL(N,Mu,Mv,Um,Vm,mu)
2056. tic;
2058. M = Mu\*Mv;
2060. Up= Um/8; % ШИРИНА ДН ВСЕЙ РЕШЕТКИ ПО ОДНОЙ ОСИ
2061. Vp= Vm/8; % ШИРИНА ДН ВСЕЙ РЕШЕТКИ ПО ДРУГОЙ ОСИ
2062. %-----------------------------------------------------
2063. lu=6; % КОЛИЧЕСТВО СЕКТОРОВ ПО ОДНОЙ КООРДИНАТЕ
2064. lv=6; % КОЛИЧЕСТВО СЕКТОРОВ ПО ДРУГОЙ КООРДИНАТЕ
2065. lt=16; % ОПРЕДЕЛЯЕТСЯ ДАЛЬНОСТЬЮ ПРОСМОТРА
2067. sig=0; % ТИП СИГНАЛА 0=импульс 1=ЛЧМ
2068. f=1; % ТИП ПРОСМОТРА ПО ВРЕМЕНИ, ТОЛЬКО ДЛЯ ИМПУЛЬСНОГО СИГНАЛА 1= через ls отсчетов 0=через 1 отсчет
2069. if sig
2070. %---------------------------------------------------
2071. ls=50;
2072. Kc=50;
2073. ll=0:.1:N-ls;
2074. Ns=length(ll);
2075. else
2076. Kc=0;
2077. ls=9;
2079. if f
2080. % ДЛИТЕЛЬНОСТЬ СИГНАЛА
2081. Ns=fix(N/ls); % КОЛИЧЕССТВО ПРОВЕРЯЕМЫХ ГИПОТЕЗ ПО ВРЕМЕНИ
2082. ll=1:ls:N-ls; % ЗНАЧЕНИЯ НАЧАЛА СИГНАЛА В ГИПОТЕЗАХ
2083. else
2084. Ns=N-ls;
2085. ll=1:N-ls;
2086. end
2087. end
2089. U\_=1\*Up; % ШАГ РАЗБИВКИ ПРОСТРАНСТВА В ДОЛЯХ ОТ ШИРИНЫ ДН РЕШЕТКИ
2090. ii=0:U\_:Um/2;
2091. ii=[-ii(end:-1:2) ii]; % ЗНАЧЕНИЯ ОДНОГО УГЛА В ГИПОТЕЗАХ
2092. Nu=length(ii); % КОЛИЧЕСТВО ПРОВЕРЯЕМЫХ ГИПОТЕЗ ПО ОДНОМУ УГЛУ
2093. %-----------------------------------------------------
2094. V\_=1\*Vp; % ШАГ РАЗБИВКИ ПРОСТРАНСТВА В ДОЛЯХ ОТ ШИРИНЫ ДН РЕШЕТКИ
2095. kk=0:V\_:Vm/2;
2096. kk=[-kk(end:-1:2) kk]; % ЗНАЧЕНИЯ ДРУГОГО УГЛА В ГИПОТЕЗАХ
2097. Nv=length(kk); % КОЛИЧЕСТВО ПРОВЕРЯЕМЫХ ГИПОТЕЗ ПО ДРУГОМУ УГЛУ
2099. L=zeros(Nu\*Nv,Ns);
2101. load Sx.txt;
2102. load Sy.txt;
2104. S= reshape(Sx,N,Ns)+j\*reshape(Sy,N,Ns);
2106. load Ux.txt;
2107. load Uy.txt;
2109. U= reshape(Ux,M,Nu\*Nv)+j\*reshape(Uy,M,Nu\*Nv);
2111. load YX.txt;
2112. load YY.txt;
2114. Y= reshape(YX,Mu\*Mv,N,lu,lv,lt)+j\*reshape(YY,Mu\*Mv,N,lu,lv,lt);
2116. load Rx.txt;
2117. load Ry.txt;
2119. Rinv=reshape(Rx,M,M,lu,lv,lt)+j\*reshape(Ry,M,M,lu,lv,lt);
2121. for i=1:lu
2122. for k=1:lv
2123. for l=1:lt
2124. YS=Y(:,:,i,k,l)\*S;
2125. RYS=Rinv(:,:,i,k,l)\*YS;
2126. c1=abs(sum(U'.'.\*(Rinv(:,:,i,k,l)\*U)).');
2127. c2=abs(1-sum(YS'.'.\*RYS));
2128. L(:,:,i,k,l)=abs(U'\*RYS).^2./(c1\*c2);
2129. L(:,:,i,k,l)=real(L(:,:,i,k,l));
2130. L(:,:,i,k,l)=mu\*L(:,:,i,k,l)./(1-L(:,:,i,k,l)); % ФОРМИРОВАНИЕ ТРЕБУЕМОЙ СТАТИСТИКИ
2131. end
2132. end
2133. end
2135. Lfile=L(:,:);
2136. save Lfile.txt Lfile -ascii;
2138. Toc = toc;
2139. end
      * 1. Код функции CreatePicture.m
2140. function Toc=CreatePicture(N,Um,Vm, H)
2141. tic;
2143. Up= Um/8; % ШИРИНА ДН ВСЕЙ РЕШЕТКИ ПО ОДНОЙ ОСИ
2144. Vp= Vm/8; % ШИРИНА ДН ВСЕЙ РЕШЕТКИ ПО ДРУГОЙ ОСИ
2145. %-----------------------------------------------------
2146. lu=6; % КОЛИЧЕСТВО СЕКТОРОВ ПО ОДНОЙ КООРДИНАТЕ
2147. lv=6; % КОЛИЧЕСТВО СЕКТОРОВ ПО ДРУГОЙ КООРДИНАТЕ
2148. lt=16; % ОПРЕДЕЛЯЕТСЯ ДАЛЬНОСТЬЮ ПРОСМОТРА
2149. %--------------------------------------------
2150. sig=0; % ТИП СИГНАЛА 0=импульс 1=ЛЧМ
2151. f=1; % ТИП ПРОСМОТРА ПО ВРЕМЕНИ, ТОЛЬКО ДЛЯ ИМПУЛЬСНОГО СИГНАЛА 1= через ls отсчетов 0=через 1 отсчет
2152. if sig
2153. %---------------------------------------------------
2154. ls=50;
2155. Kc=50;
2156. ll=0:.1:N-ls;
2157. Ns=length(ll);
2158. else
2159. Kc=0;
2160. ls=9;
2162. if f
2163. % ДЛИТЕЛЬНОСТЬ СИГНАЛА
2164. Ns=fix(N/ls); % КОЛИЧЕССТВО ПРОВЕРЯЕМЫХ ГИПОТЕЗ ПО ВРЕМЕНИ
2165. ll=1:ls:N-ls; % ЗНАЧЕНИЯ НАЧАЛА СИГНАЛА В ГИПОТЕЗАХ
2166. else
2167. Ns=N-ls;
2168. ll=1:N-ls;
2169. end
2170. end
2172. U\_=1\*Up; % ШАГ РАЗБИВКИ ПРОСТРАНСТВА В ДОЛЯХ ОТ ШИРИНЫ ДН РЕШЕТКИ
2173. ii=0:U\_:Um/2;
2174. ii=[-ii(end:-1:2) ii]; % ЗНАЧЕНИЯ ОДНОГО УГЛА В ГИПОТЕЗАХ
2175. Nu=length(ii); % КОЛИЧЕСТВО ПРОВЕРЯЕМЫХ ГИПОТЕЗ ПО ОДНОМУ УГЛУ
2176. %-----------------------------------------------------
2177. V\_=1\*Vp; % ШАГ РАЗБИВКИ ПРОСТРАНСТВА В ДОЛЯХ ОТ ШИРИНЫ ДН РЕШЕТКИ
2178. kk=0:V\_:Vm/2;
2179. kk=[-kk(end:-1:2) kk]; % ЗНАЧЕНИЯ ДРУГОГО УГЛА В ГИПОТЕЗАХ
2180. Nv=length(kk); % КОЛИЧЕСТВО ПРОВЕРЯЕМЫХ ГИПОТЕЗ ПО ДРУГОМУ УГЛУ
2182. load Lfile.txt;
2183. L = Lfile;
2185. figure(1);
2186. Hm=max(L(:));
2187. %Hm=90;
2188. %L2=L(:,:,1,1,1);
2189. L2=L;
2190. ind=find(L2>H);
2191. v=L2(L2>H);
2192. vvv=v;
2194. % [x1,y1,z1]=ind2sub([Nu Nv Ns],ind);
2195. [x1,y1,z1,u1,v1,t1]=ind2sub([Nu Nv Ns lu lv lt],ind);
2196. % x=ii(x1);
2197. % y=kk(y1);
2198. % z=ll(z1);
2200. x=ii(x1)'+(u1-1)\*Um;
2201. y=kk(y1)'+(v1-1)\*Vm;
2202. z=ll(z1)'+(t1-1)\*N;
2204. delete(gca)
2205. marker='o';
2206. string='сигнал';
2207. miv=H;
2208. mav=Hm;
2209. % Get the current colormap
2210. map=colormap;
2212. hold on
2213. for i=1:length(x)
2214. in=round((v(i)-miv)\*(length(map)-1)/(mav-miv));
2215. %--- Catch the out-of-range numbers
2216. if in==0;in=1;end
2217. if in > length(map);in=length(map);end
2218. plot3(x(i),y(i),z(i),marker,'color',map(in,:),'markerfacecolor',map(in,:))
2220. end
2221. hold off
2223. % Re-format the colorbar
2224. h=colorbar;
2226. %set(h,'ylim',[1 length(map)]);
2227. yal=linspace(1,length(map),10);
2228. %set(h,'ytick',yal);
2230. % Create the yticklabels
2231. ytl=linspace(miv,mav,11);
2233. s=char(11,4);
2235. for i=1:11
2236. if abs(min(log10(abs(ytl)))) <= 3
2237. B=sprintf('%-4.3f',ytl(i));
2238. else
2239. B=sprintf('%-4.2E',ytl(i));
2240. end
2241. s(i,1:length(B))=B;
2242. end
2244. set(h,'yticklabel',s,'fontsize',9);
2245. grid on
2246. set(get(h,'title'),'string',string)
2247. view(3)
2249. Toc=toc;
2250. end
      * 1. Код функции getGauss.m
2251. function C=getGauss(M,N)
2252. p=54;
2253. cc=1/3;
2254. c1=sum(rand(M,N,p),3);
2255. c2=sum(rand(M,N,p),3);
2256. C=cc\*(c1-p/2)+j\*cc\*(c2-p/2);
2257. end
      * 1. Код функции getMatrNabl.m
2258. function Y=getMatrNabl(N,Mu,Mv,Um,Vm,u,v,a,Tc,Lc,Kc,up,vp,Ap,gamma)
2259. M=Mu\*Mv;
2260. Kp=length(up);
2261. Nc=length(u);
2263. g=sinx(pi\*u/Um).\*sinx(pi\*v/Vm);
2264. f=getPhase(u/Um,v/Vm,Mu,Mv);
2265. U=[repmat(g,M,1).\*exp(f)];
2266. S=zeros(N,Nc);
2267. n=1:N;
2268. for l=1:Nc
2269. b=n-1>=Tc(l)&n-1<Tc(l)+Lc;
2270. o=(n(b)-1-Tc(l))/Lc;
2271. S(n(b),l)=exp(-j\*pi\*Kc\*(o.^2-o));
2272. end
2273. g=sinx(pi\*up/Um).\*sinx(pi\*vp/Vm);
2274. f=getPhase(up/Um,vp/Vm,Mu,Mv);
2275. V=[repmat(g,M,1).\*exp(f) repmat(g,M,1).\*f.\*exp(f)];
2276. E=[repmat(Ap,N,1).\*getGauss(N,Kp), repmat(Ap.\*gamma,N,1).\*getGauss(N,Kp)];
2277. Y=V\*E'+(U.\*repmat(a,M,1))\*S'+getGauss(M,N);
2278. end
      * 1. Код функции getPhase.m
2279. function F=getPhase(u,v,Mu,Mv,Um,Vm)
2280. [p q]=ind2sub([Mu Mv],1:Mu\*Mv);
2281. F=2\*j\*pi\*[p' q']\*[u;v];
2282. end
      * 1. Код функции sinx.m
2283. function r=sinx(x)
2284. r=ones(size(x));
2285. c=x~=0;
2286. r(c)=sin(x(c))./x(c);
      * 1. Код функции Yinvert.m
2287. function Ri = Yinvert(Y,N,M,mu)
2288. %R = mu\*eye(N,N) + Y\*(Y')
2289. Ri = eye(M,M)./mu;
2290. for j = 1:N
2291. Ry = Ri\*Y(:,j);
2292. Ri = Ri - ( Ry\*Ry' )./( 1 + Y(:,j)'\*Ry );
2293. end

# СПИСОК ЛИТЕРАТУРЫ

1. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 15150-69 Машины, приборы и другие технические изделия. Исполнения для различных климатических районов. Категории, условия эксплуатации, хранения и транспортирования в части воздействия климатических факторов внешней среды. – М.: Изд-во стандартов, 1997.
3. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.404-79. ЕСПД. Пояснительная записка. Требования к содержанию и оформлению. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.505-79. ЕСПД. Руководство оператора. Требования к содержанию и оформлению. . – М.: ИПК Издательство стандартов, 2001.
7. Черемисин О.П., Пантюхин Д.В., Подложнюк В.Д. Оценка технической эффективности применения современных графических ускорителей в задаче обнаружения целей на фоне помех в автоматизированных фазированных антенных решетках // Информатизация и связь. 2009, № 1, С. 151–153.

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | Новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |