**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет прикладной математики и информационных технологий

Кафедра вычислительной математики и программирования

**Отчет по лабораторным работам**

по курсу «Численные Методы»

Студент: Иларионов Д.А.

Группа: М8О-408Б-17

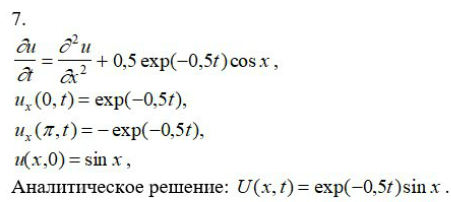
Преподаватель: Ревизников Д.Л.

**Лабораторная работа 5**

1. Тема ЛР:

Численное решение уравнений параболического типа. Понятие о методе конечных разностей. Основные определения и конечно-разностные схемы. Решить уравнение параболического типа с помощью явной, неявной схемы и схемы Кранка-Никольсона.

1. Вариант : **7**



1. Алгоритм:

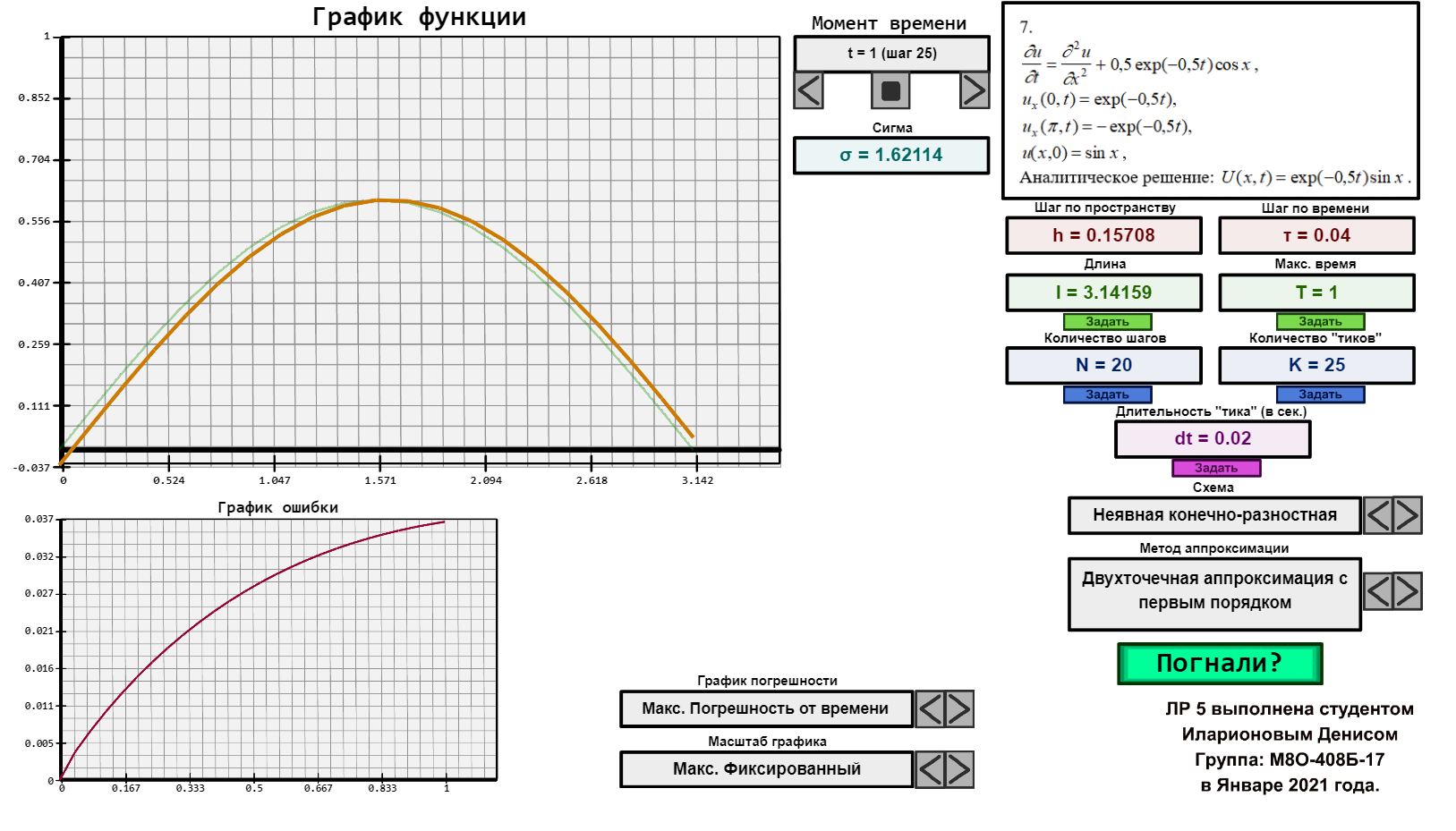
Необходимо было решить уравнение мат. физики параболического тема с помощью конечно-разностных схем. Так как урматы я плохо понимал, ушло очень много времени на то, чтобы мне удалось что-то понять. В итоге, я смог реализовать алгоритмы. Трудной частью лабораторной также была аппроксимация граничных условий. С явной схемой все понятно, мы просто выводим формулу, потому что используются элементы прошлого временного слоя. С неявной немного труднее – у нас 3 неизвестных на каждой итерации. Но это можно решить с помощью метода прогонки, создав новую матрицу и подставив коэффициенты. С этим у меня была проблема и программа совсем плохо генерировала график. Кстати, самое главное – мы все уравнение разбиваем на блоки на двухмерной сетке. Такой принцип будет во всех лабораторных (а в последней и вовсе трехмерная сетка). Схема Кранка-Никольсона является комбинацией явной и неявной схемы и иногда дает лучше результаты, потому что явная схема часто опережает уравнение, а неявная – наоборот. Да, еще, явная схема является устойчивой, только тогда, когда число сигма (a\*a\*t/h^2) < 0.5. Где a – коэффициент перед производной. t – размер шага по времени, а h – размер шага по пространству. Неявная схема же всегда устойчива.

1. Среда разработки:

Adobe Animate, Javascript + HTML5

1. Реализация

Нарисовал и реализовал интерфейс для работы с уравнением. Можно задать шаги, скорость шага. Можно посмотреть погрешность, а также менять масштаб графика (динамический или статический).



1. Код (javascript)
2. **var** alpha = 1; *//α*
3. **var** beta = 0; *//β*
4. **var** gamma = 1; *//γ*
5. **var** delta = 0; *//δ*

8. *//α \* Ux(0, t) + β \* U(0, t) = exp(-0.5\*t)*
9. *//γ \* Ux(l, t) + δ \* U(l, t) = -exp(-0.5\*t)*
10. *//Обозначим все как граничные условия 3 рода*
12. **function** phi\_0(t) {
13. **return** Math.exp(-0.5 \* t);
14. }
16. **function** phi\_l(t) {
17. **return** -Math.exp(-0.5 \* t);
18. }
20. **function** psi(x) { **return** Math.sin(x); }
21. *// Начальное условие*


25. **function** U(x, t) {
26. **return** Math.exp(-0.5 \* t) \* Math.sin(x);
27. } *//Аналитическое решение*

30. *//Параметры сетки*
31. **var** l = 3.1415926535; *//интервал*
32. **var** N = 10; *//количество шагов*
33. **var** T = 1; *//макс. время*
34. **var** K = 25; *//количество тиков*
36. **var** h = 0; **var** t = 0;
38. **var** a = 1; **var** b = 0; **var** c = 0;
39. *//exp(-0.5t)cos(x) - доп. функция*
41. **var** sigma = 1;
42. *//σ = a^2 \* τ / h^2*
44. **var** dt = 0.02; *//длительность тика*
45. **var** cur\_tick = 0;

48. **var** generation = 0; *//номер генерации*
49. **var** go = **false**; *//увеличивать счетчик тиков?*

52. **this**.addEventListener("tick", det\_ht.bind(**this**));
53. **function** det\_ht() {
54. h = l / N;
55. t = T / K;
57. sigma = a \* a \* t / (h \* h);
58. *//определяем шаги по времени и пространству*
60. **this**.h\_text.text = "h = " + Math.round(h \* 100000)/100000;
61. **this**.tau\_text.text = "τ = " + Math.round(t \* 100000)/100000;
63. **this**.l\_text.text = "l = " + Math.round(l \* 100000)/100000;
64. **this**.T\_text.text = "T = " + Math.round(T \* 100000)/100000;
66. **this**.N\_text.text = "N = " + Math.round(N);
67. **this**.K\_text.text = "K = " + Math.round(K);
69. **this**.dt\_text.text = "dt = " + Math.round(dt\*1000)/1000;
71. **this**.sig\_text.text = "σ = " + Math.round(sigma \* 100000)/100000;

74. **if** (go) {
75. cur\_tick += 1/60;
76. **if** (cur\_tick >= dt && s\_cur < K) {
77. **var** tGot = Math.floor(cur\_tick / dt);
78. cur\_tick %= dt;
79. s\_cur += Math.min(K - s\_cur, tGot);
80. }
81. **else** **if** (s\_cur == K) {
82. go = **false**;
83. }
84. }
86. }



91. **this**.setGreen1.addEventListener("click", set\_l\_prompt.bind(**this**));
92. **function** set\_l\_prompt() {
93. generation += 1;
94. **var** temp = prompt("Введите l:", '');
95. temp = Number.parseFloat(temp);
96. **if** (isNaN(temp)) {
97. *//если ввели какую-то хрень*
98. temp = l;
99. }
100. **else** **if** (temp > 1000000) {
101. *//ограничение на величину*
102. temp = 1000000;
103. }
104. **else** **if** (temp <= 0) {
105. *//всегда положительно*
106. temp = 0.001;
107. }
108. l = temp;
109. }


113. **this**.setGreen2.addEventListener("click", set\_T\_prompt.bind(**this**));
114. **function** set\_T\_prompt() {
115. generation += 1;
116. **var** temp = prompt("Введите T:", '');
117. temp = Number.parseFloat(temp);
118. **if** (isNaN(temp)) {
119. *//если ввели какую-то хрень*
120. temp = T;
121. }
122. **else** **if** (temp > 1000000) {
123. *//ограничение на величину*
124. temp = 1000000;
125. }
126. **else** **if** (temp <= 0) {
127. *//всегда положительно*
128. temp = 0.001;
129. }
130. T = temp;
131. }

134. **this**.setBlue1.addEventListener("click", set\_N\_prompt.bind(**this**));
135. **function** set\_N\_prompt() {
136. generation += 1;
137. **var** temp = prompt("Введите N:", '');
138. temp = Number.parseInt(temp);
139. **if** (isNaN(temp)) {
140. *//если ввели какую-то хрень*
141. temp = N;
142. }
143. **else** **if** (temp > 200) {
144. *//ограничение на величину*
145. temp = 200;
146. }
147. **else** **if** (temp <= 0) {
148. *//всегда положительно*
149. temp = 1;
150. }
151. N = temp;
152. }

155. **this**.setBlue2.addEventListener("click", set\_K\_prompt.bind(**this**));
156. **function** set\_K\_prompt() {
157. generation += 1;
158. **var** temp = prompt("Введите K:", '');
159. temp = Number.parseInt(temp);
160. **if** (isNaN(temp)) {
161. *//если ввели какую-то хрень*
162. temp = K;
163. }
164. **else** **if** (temp > 10000) {
165. *//ограничение на величину*
166. temp = 10000;
167. }
168. **else** **if** (temp <= 0) {
169. *//всегда положительно*
170. temp = 1;
171. }
172. K = temp;
173. }

176. **this**.setPurp1.addEventListener("click", set\_dt\_prompt.bind(**this**));
177. **function** set\_dt\_prompt() {
178. cur\_tick = 0;
179. **var** temp = prompt("Введите dt (для ручного переключения можно ввести очень большим):", '');
180. temp = Number.parseFloat(temp);
181. **if** (isNaN(temp)) {
182. *//если ввели какую-то хрень*
183. temp = dt;
184. }
185. **else** **if** (temp > 1000000) {
186. *//ограничение на величину*
187. temp = 1000000;
188. }
189. **else** **if** (temp < 0.02) {
190. *//всегда положительно и больше 0.2 (почти плавная смена графика)*
191. temp = 0.02;
192. }
193. dt = temp;
194. }


198. **var** scheme\_type = 1;
199. *//Тип схемы*
200. *//1,2 - Явная/Неявная К-Р схема*
201. *//3 - схема Кранка-Николсона*
203. **var** theta = 1;

206. **var** meth\_type = 1;
207. *//Метод аппроксимации*
208. *//1 - Двухточечная аппроксимация с первым порядком*
209. *//2 - Трехточечная аппроксимация со вторым порядком*
210. *//3 - Двухточечная аппроксимация со вторым порядком*

213. **var** pogr\_type = 1;
214. *//Тип погрешности*
215. *//1 - Абсолютная*
216. *//2 - Относительная*
217. *//3 - От времени*
219. **var** scale\_type = 1;
220. *//Масштаб графика*
221. *//1 - Фиксированный*
222. *//2 - Динамический*
224. **this**.SLeft1.addEventListener("click", scheme\_left.bind(**this**));
225. **function** scheme\_left() {
226. **if** (scheme\_type == 1) {
227. scheme\_type = 3;
228. }
229. **else** {
230. scheme\_type -= 1;
231. }
232. }
234. **this**.SRight1.addEventListener("click", scheme\_right.bind(**this**));
235. **function** scheme\_right() {
236. **if** (scheme\_type == 3) {
237. scheme\_type = 1;
238. }
239. **else** {
240. scheme\_type += 1;
241. }
242. }

245. **this**.SLeft2.addEventListener("click", meth\_left.bind(**this**));
246. **function** meth\_left() {
247. **if** (meth\_type == 1) {
248. meth\_type = 3;
249. }
250. **else** {
251. meth\_type -= 1;
252. }
253. }
255. **this**.SRight2.addEventListener("click", meth\_right.bind(**this**));
256. **function** meth\_right() {
257. **if** (meth\_type == 3) {
258. meth\_type = 1;
259. }
260. **else** {
261. meth\_type += 1;
262. }
263. }

266. **this**.SLeft3.addEventListener("click", curStep\_left.bind(**this**));
267. **function** curStep\_left() {
268. **if** (s\_cur > 0) {
269. s\_cur -= 1;
270. }
271. }
273. **this**.SRight3.addEventListener("click", curStep\_right.bind(**this**));
274. **function** curStep\_right() {
275. **if** (s\_cur < K) {
276. s\_cur += 1;
277. }
278. }

281. **this**.stopBtn.addEventListener("click", curStep\_stop.bind(**this**));
282. **function** curStep\_stop() {
283. dt = 1000000;
284. }
286. **this**.SLeft4.addEventListener("click", pogr\_left.bind(**this**));
287. **function** pogr\_left() {
288. **if** (pogr\_type == 1) {
289. pogr\_type = 3;
290. }
291. **else** {
292. pogr\_type -= 1;
293. }
294. }
296. **this**.SRight4.addEventListener("click", pogr\_right.bind(**this**));
297. **function** pogr\_right() {
298. **if** (pogr\_type == 3) {
299. pogr\_type = 1;
300. }
301. **else** {
302. pogr\_type += 1;
303. }
304. }
306. **this**.SLeft5.addEventListener("click", scale\_left.bind(**this**));
307. **function** scale\_left() {
308. **if** (scale\_type == 1) {
309. scale\_type = 2;
310. }
311. **else** {
312. scale\_type -= 1;
313. }
314. }
316. **this**.SRight5.addEventListener("click", scale\_right.bind(**this**));
317. **function** scale\_right() {
318. **if** (scale\_type == 2) {
319. scale\_type = 1;
320. }
321. **else** {
322. scale\_type += 1;
323. }
324. }
326. **this**.addEventListener("tick", setTexts2.bind(**this**));
327. **function** setTexts2() {
328. **if** (scheme\_type == 1) {
329. **this**.scheme\_text.text = "Явная конечно-разностная";
330. theta = 1;
331. }
332. **else** **if** (scheme\_type == 2) {
333. **this**.scheme\_text.text = "Неявная конечно-разностная";
334. theta = 0;
335. }
336. **else** **if** (scheme\_type == 3) {
337. **this**.scheme\_text.text = "Схема Кранка-Николсона";
338. theta = 0.5;
339. }
341. **if** (meth\_type == 1) {
342. **this**.meth\_text.text = "Двухточечная аппроксимация с первым порядком";
343. }
344. **else** **if** (meth\_type == 2) {
345. **this**.meth\_text.text = "Трехточечная аппроксимация со вторым порядком";
346. }
347. **else** **if** (meth\_type == 3) {
348. **this**.meth\_text.text = "Двухточечная аппроксимация со вторым порядком";
349. }

352. **if** (pogr\_type == 1) {
353. **this**.pogr\_text.text = "Абсолютная погрешность";
354. }
355. **else** **if** (pogr\_type == 2) {
356. **this**.pogr\_text.text = "Относительная погрешность";
357. }
358. **else** **if** (pogr\_type == 3) {
359. **this**.pogr\_text.text = "Макс. Погрешность от времени";
360. }

363. **if** (scale\_type == 1) {
364. **this**.scale\_text.text = "Макс. Фиксированный";
365. }
366. **else** **if** (scale\_type == 2) {
367. **this**.scale\_text.text = "Динамический";
368. }

371. **this**.divDown0.text = "" + 0;
372. **this**.divDown1.text = "" + Math.round(l\*1/6\*1000)/1000;
373. **this**.divDown2.text = "" + Math.round(l\*2/6\*1000)/1000;
374. **this**.divDown3.text = "" + Math.round(l\*3/6\*1000)/1000;
375. **this**.divDown4.text = "" + Math.round(l\*4/6\*1000)/1000;
376. **this**.divDown5.text = "" + Math.round(l\*5/6\*1000)/1000;
377. **this**.divDown6.text = "" + Math.round(l\*1000)/1000;

380. **if** (pogr\_type <= 2) {
381. **this**.divDown0err.text = "" + 0;
382. **this**.divDown1err.text = "" + Math.round(l\*1/6\*1000)/1000;
383. **this**.divDown2err.text = "" + Math.round(l\*2/6\*1000)/1000;
384. **this**.divDown3err.text = "" + Math.round(l\*3/6\*1000)/1000;
385. **this**.divDown4err.text = "" + Math.round(l\*4/6\*1000)/1000;
386. **this**.divDown5err.text = "" + Math.round(l\*5/6\*1000)/1000;
387. **this**.divDown6err.text = "" + Math.round(l\*1000)/1000;
388. }
389. **else** {
390. **this**.divDown0err.text = "" + 0;
391. **this**.divDown1err.text = "" + Math.round(T\*1/6\*1000)/1000;
392. **this**.divDown2err.text = "" + Math.round(T\*2/6\*1000)/1000;
393. **this**.divDown3err.text = "" + Math.round(T\*3/6\*1000)/1000;
394. **this**.divDown4err.text = "" + Math.round(T\*4/6\*1000)/1000;
395. **this**.divDown5err.text = "" + Math.round(T\*5/6\*1000)/1000;
396. **this**.divDown6err.text = "" + Math.round(T\*1000)/1000;
397. }

400. **if** (maxVals.length > 0 && scale\_type == 2) {
401. **this**.divUp0.text = "" + Math.round(minVals[s\_cur]\*1000)/1000;
402. **this**.divUp1.text = "" + Math.round((minVals[s\_cur] + (maxVals[s\_cur] - minVals[s\_cur])\*1/7)\*1000)/1000;
403. **this**.divUp2.text = "" + Math.round((minVals[s\_cur] + (maxVals[s\_cur] - minVals[s\_cur])\*2/7)\*1000)/1000;
404. **this**.divUp3.text = "" + Math.round((minVals[s\_cur] + (maxVals[s\_cur] - minVals[s\_cur])\*3/7)\*1000)/1000;
405. **this**.divUp4.text = "" + Math.round((minVals[s\_cur] + (maxVals[s\_cur] - minVals[s\_cur])\*4/7)\*1000)/1000;
406. **this**.divUp5.text = "" + Math.round((minVals[s\_cur] + (maxVals[s\_cur] - minVals[s\_cur])\*5/7)\*1000)/1000;
407. **this**.divUp6.text = "" + Math.round((minVals[s\_cur] + (maxVals[s\_cur] - minVals[s\_cur])\*6/7)\*1000)/1000;
408. **this**.divUp7.text = "" + Math.round(maxVals[s\_cur]\*1000)/1000;
409. }
410. **else** {
411. **this**.divUp0.text = "" + Math.round(minMinVal\*1000)/1000;
412. **this**.divUp1.text = "" + Math.round((minMinVal + (maxMaxVal - minMinVal)\*1/7)\*1000)/1000;
413. **this**.divUp2.text = "" + Math.round((minMinVal + (maxMaxVal - minMinVal)\*2/7)\*1000)/1000;
414. **this**.divUp3.text = "" + Math.round((minMinVal + (maxMaxVal - minMinVal)\*3/7)\*1000)/1000;
415. **this**.divUp4.text = "" + Math.round((minMinVal + (maxMaxVal - minMinVal)\*4/7)\*1000)/1000;
416. **this**.divUp5.text = "" + Math.round((minMinVal + (maxMaxVal - minMinVal)\*5/7)\*1000)/1000;
417. **this**.divUp6.text = "" + Math.round((minMinVal + (maxMaxVal - minMinVal)\*6/7)\*1000)/1000;
418. **this**.divUp7.text = "" + Math.round(maxMaxVal\*1000)/1000;
419. }
421. **if** (maxValsE.length > 0 && scale\_type == 2 && pogr\_type <= 2) {
422. **this**.divUp0err.text = "" + Math.round(minValsE[s\_cur]\*1000)/1000;
423. **this**.divUp1err.text = "" + Math.round((minValsE[s\_cur] + (maxValsE[s\_cur] - minValsE[s\_cur])\*1/7)\*1000)/1000;
424. **this**.divUp2err.text = "" + Math.round((minValsE[s\_cur] + (maxValsE[s\_cur] - minValsE[s\_cur])\*2/7)\*1000)/1000;
425. **this**.divUp3err.text = "" + Math.round((minValsE[s\_cur] + (maxValsE[s\_cur] - minValsE[s\_cur])\*3/7)\*1000)/1000;
426. **this**.divUp4err.text = "" + Math.round((minValsE[s\_cur] + (maxValsE[s\_cur] - minValsE[s\_cur])\*4/7)\*1000)/1000;
427. **this**.divUp5err.text = "" + Math.round((minValsE[s\_cur] + (maxValsE[s\_cur] - minValsE[s\_cur])\*5/7)\*1000)/1000;
428. **this**.divUp6err.text = "" + Math.round((minValsE[s\_cur] + (maxValsE[s\_cur] - minValsE[s\_cur])\*6/7)\*1000)/1000;
429. **this**.divUp7err.text = "" + Math.round(maxValsE[s\_cur]\*1000)/1000;
430. }
431. **else** **if** (pogr\_type <= 2) {
432. **this**.divUp0err.text = "" + Math.round(minMinValE\*1000)/1000;
433. **this**.divUp1err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*1/7)\*1000)/1000;
434. **this**.divUp2err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*2/7)\*1000)/1000;
435. **this**.divUp3err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*3/7)\*1000)/1000;
436. **this**.divUp4err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*4/7)\*1000)/1000;
437. **this**.divUp5err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*5/7)\*1000)/1000;
438. **this**.divUp6err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*6/7)\*1000)/1000;
439. **this**.divUp7err.text = "" + Math.round(maxMaxValE\*1000)/1000;
440. }
441. **else** {
442. **this**.divUp0err.text = "" + Math.round(minEotT\*1000)/1000;
443. **this**.divUp1err.text = "" + Math.round((minEotT + (maxEotT - minEotT)\*1/7)\*1000)/1000;
444. **this**.divUp2err.text = "" + Math.round((minEotT + (maxEotT - minEotT)\*2/7)\*1000)/1000;
445. **this**.divUp3err.text = "" + Math.round((minEotT + (maxEotT - minEotT)\*3/7)\*1000)/1000;
446. **this**.divUp4err.text = "" + Math.round((minEotT + (maxEotT - minEotT)\*4/7)\*1000)/1000;
447. **this**.divUp5err.text = "" + Math.round((minEotT + (maxEotT - minEotT)\*5/7)\*1000)/1000;
448. **this**.divUp6err.text = "" + Math.round((minEotT + (maxEotT - minEotT)\*6/7)\*1000)/1000;
449. **this**.divUp7err.text = "" + Math.round(maxEotT\*1000)/1000;
450. }
452. t\_cur = s\_cur\*t;
454. **this**.step\_text.text = "t = " + Math.round(t\_cur\*10000)/10000 + " (шаг " + Math.round(s\_cur) + ")";
456. **this**.axisDown.y = Math.min(Math.max(transfY(0), 26.4), 418.6);
458. **this**.axisDownErr.y = Math.min(Math.max(transfEY(0), 468), 700);
459. }

462. *//baseGraph*
463. *//x = 0 : posX = 60*
464. *//y = 0 : posY = 418.6*
466. *//x = max : posX = 627*
467. *//y = max : posY = 26.4*
469. *//errGraph*
470. *//x = 0 : posX = 53.6*
471. *//y = 0 : posY = 700*
473. *//x = max : posX = 397.8*
474. *//y = max : posY = 468.4*




480. **var** t\_cur = 0;
481. *//текущий момент времени*
483. **var** s\_cur = 0;
484. *//текущий шаг*

487. **var** etalon = [];
488. **var** xVector = [];
489. **var** tVector = [];
491. **var** maxMaxVal = 1;
492. **var** minMinVal = 0;
494. **var** maxMaxValE = 1; *//link*
495. **var** minMinValE = 0; *//link*
496. **var** maxMaxValE1 = 1;
497. **var** minMinValE1 = 0;
498. **var** maxMaxValE2 = 1;
499. **var** minMinValE2 = 0;
501. **var** maxVals = [];
502. **var** minVals = [];
504. **var** maxValsE = []; *//link*
505. **var** minValsE = []; *//link*
506. **var** maxValsE1 = [];
507. **var** minValsE1 = [];
508. **var** maxValsE2 = [];
509. **var** minValsE2 = [];

512. **var** EotT = [];
514. **this**.addEventListener("tick", setError.bind(**this**));
515. **function** setError() {
516. **if** (pogr\_type == 1) {
517. maxMaxValE = maxMaxValE1;
518. minMinValE = minMinValE1;
519. maxValsE = maxValsE1;
520. minValsE = minValsE1;
521. errorGraph = absolute\_error;
522. }
523. **else** {
524. maxMaxValE = maxMaxValE2;
525. minMinValE = minMinValE2;
526. maxValsE = maxValsE2;
527. minValsE = minValsE2;
528. errorGraph = relative\_error;
529. }
530. }
532. **var** solve = [];

535. **function** copyV(vector) {
536. **var** newV = [];
537. **for** (**var** i = 0 ; i < vector.length ; ++i) {
538. newV.push(vector[i]);
539. }
540. **return** newV;
541. }

544. **var** absolute\_error = [];
545. **var** relative\_error = [];
546. **var** errorGraph = [];
548. **var** minEotT = 0;
549. **var** maxEotT = 1;


553. **function** explicit\_step(i, vector0, vector1) {
554. **var** curT = i \* t;
556. *//Сначала вычисляем серединку*
557. **for** (**var** j = 1 ; j < N ; ++j) {
558. vector1[j] = sigma \* vector0[j + 1] + (1 - 2\*sigma) \* vector0[j] + sigma \* vector0[j - 1] + 0.5\*Math.exp(-0.5\*curT)\*Math.sin((j-1)\*h)\*t;
560. maxVals[i] = (vector1[j] > maxVals[i] ? vector1[j] : maxVals[i]);
561. minVals[i] = (vector1[j] < minVals[i] ? vector1[j] : minVals[i]);
562. }
564. *//аппроксимируем граничные условия*
565. **if** (meth\_type == 1) {
566. vector1[0] = vector1[1] - (h/alpha)\*phi\_0(curT);
567. vector1[N] = vector1[N-1] + (h/gamma)\*phi\_l(curT);
568. }
569. **if** (meth\_type == 2) {
570. *//(u1k - u0k)/h = (-3u0k + 4u1k - u2k)/2h*
571. *//вывожу u0k*
572. *//получаю u0k = 2u1k - u2k*
573. *//аналогично для ulk*
574. vector1[0] = (4 \* vector1[1] - vector1[2] - 2\*h\*phi\_0(curT))/3;
575. vector1[N] = (4 \* vector1[N-1] - vector1[N-2] + 2\*h\*phi\_l(curT))/3;
576. }
577. **if** (meth\_type == 3) {
578. *//тут идут сложные преобразования, которые*
579. *//я делал на бумаге*
580. **var** d0 = h\*vector0[0]/t - (phi\_0(curT)\*(2\*a\*a/alpha));
581. **var** c0 = -2 \* a \* a / h;
582. **var** b0 = (2 \* a \* a / h) + (h/t);
583. **var** a0 = 0;
585. vector1[0] = (d0 - c0\*vector1[1]) / b0;
587. **var** dn = h\*vector0[N]/t + (phi\_l(curT)\*(2\*a\*a/gamma));
588. **var** cn = 0;
589. **var** bn = 2\*a\*a/h + (h/t);
590. **var** an = -2\*a\*a/h;
592. vector1[N] = (dn - an\*vector1[N-1])/bn;
593. }
595. maxVals[i] = (vector1[0] > maxVals[i] ? vector1[0] : maxVals[i]);
596. maxVals[i] = (vector1[N] > maxVals[i] ? vector1[N] : maxVals[i]);
598. minVals[i] = (vector1[0] < minVals[i] ? vector1[0] : minVals[i]);
599. minVals[i] = (vector1[N] < minVals[i] ? vector1[N] : minVals[i]);
601. maxMaxVal = (maxVals[i] > maxMaxVal ? maxVals[i] : maxMaxVal);
602. minMinVal = (minVals[i] < minMinVal ? minVals[i] : minMinVal);
604. **return** copyV(vector1);
606. }


610. **function** implicit\_step(i, vector0, vector1) {
611. **var** curT = i \* t;
613. **var** solveMatrix = [];
614. **var** solveVector = [];
616. **var** emptV = [];
617. **for** (**var** j = 0 ; j <= N ; ++j) {
618. emptV.push(0);
619. }
621. **var** thisV = copyV(emptV);
623. **if** (meth\_type == 1) {
624. thisV[0] = -1; *//b0*
625. thisV[1] = 1; *//c0*
627. solveMatrix.push(copyV(thisV));
629. **var** d0 = h\*phi\_0(curT);
631. solveVector.push(d0); *//d0*
632. }
633. **if** (meth\_type == 2) {
634. thisV[0] = -3;
635. thisV[1] = 4;
636. thisV[2] = -1
638. solveMatrix.push(copyV(thisV));
640. **var** d0 = phi\_0(curT) \* 2 \* h;
642. solveVector.push(d0); *//d0*
643. }
644. **if** (meth\_type == 3) {
645. thisV[0] = (2\*a\*a / h) + (h/t); *//b0 g(c) и beta = 0*
646. thisV[1] = (-2\*a\*a / h);
648. solveMatrix.push(copyV(thisV));
650. **var** d0 = h/t \* vector0[0] - phi\_0(curT)\*(2\*a\*a / alpha); *//b = 0*
652. solveVector.push(d0); *//d0*
653. }
655. **for** (**var** j = 1 ; j < N ; ++j) {
656. **var** thisV = copyV(emptV);
657. thisV[j - 1] = -sigma; *//aj*
658. thisV[j] = (2\*sigma + 1); *//bj*
659. thisV[j + 1] = -sigma; *//cj*
661. solveMatrix.push(copyV(thisV));
663. **var** dj = vector0[j] + 0.5\*Math.exp(-0.5\*curT)\*Math.sin((j-1)\*h)\*t;
664. solveVector.push(dj);
666. *//убираем 3 элемент (для трехдиагональности)*
667. **if** (j == 1 && meth\_type == 2) {
668. **var** divisor = solveMatrix[0][2] / solveMatrix[1][2];
669. solveMatrix[0][0] -= solveMatrix[1][0] \* divisor;
670. solveMatrix[0][1] -= solveMatrix[1][1] \* divisor;
671. solveMatrix[0][2] = 0;
673. solveVector[0] -= solveVector[1] \* divisor;
674. }


678. }
680. **var** thisV = copyV(emptV);
682. **if** (meth\_type == 1) {
683. thisV[N-1] = -1; *//an*
684. thisV[N] = 1; *//bn*
686. solveMatrix.push(copyV(thisV));
688. **var** dn = h\*phi\_l(curT);
690. solveVector.push(dn); *//d1*
691. }
692. **if** (meth\_type == 2) {
693. thisV[N-2] = 1;
694. thisV[N-1] = -4;
695. thisV[N] = 3;
697. solveMatrix.push(copyV(thisV));
699. **var** d0 = 2\*h\*phi\_l(curT);
701. solveVector.push(d0); *//d0*
703. **var** divisor = solveMatrix[N][N-2] / solveMatrix[N-1][N-2];
704. solveMatrix[N][N] -= solveMatrix[N-1][N] \* divisor;
705. solveMatrix[N][N-1] -= solveMatrix[N-1][N-1] \* divisor;
706. solveMatrix[N][N-2] = 0;
708. solveVector[N] -= solveVector[N-1] \* divisor;
709. }
710. **if** (meth\_type == 3) {
711. thisV[N-1] = 2\*a\*a/h; *//an*
712. thisV[N] = (2\*a\*a / h) + (h/t);  *//bn g(c) и delta = 0*
714. solveMatrix.push(copyV(thisV));
716. **var** dn = h/t \* vector0[N] - phi\_l(curT)\*(2\*a\*a / gamma); *//b = 0*
718. solveVector.push(dn); *//d1*
719. }
721. **var** the\_vector = progonka(solveMatrix, solveVector);

724. **for** (**var** j = 0 ; j <= N ; ++j) {
726. maxVals[i] = (the\_vector[j] > maxVals[i] ? the\_vector[j] : maxVals[i]);
727. minVals[i] = (the\_vector[j] < minVals[i] ? the\_vector[j] : minVals[i]);
729. maxMaxVal = (maxVals[i] > maxMaxVal ? maxVals[i] : maxMaxVal);
730. minMinVal = (minVals[i] < minMinVal ? minVals[i] : minMinVal);
732. }
734. **return** copyV(the\_vector);
735. }


739. **this**.beginBtn.addEventListener("click", beginSimulation.bind(**this**));
740. **function** beginSimulation() {
741. generation += 1;
742. s\_cur = 0;
743. etalon = [];
744. solve = [];
745. maxMaxVal = 1;
746. minMinVal = 0;
748. maxVals = [];
749. minVals = [];
751. xVector = [];
752. tVector = [];
754. **for** (**var** j = 0 ; j <= N ; ++j) {
755. xVector.push(h \* j);
756. }
757. **for** (**var** i = 0 ; i <= K ; ++i) {
758. **var** timeVector = [];
759. **var** this\_t = t \* i;
760. tVector.push(this\_t);
761. **var** this\_max = 0;
762. **var** this\_min = 0;
763. **for** (**var** j = 0 ; j <= N ; ++j) {
764. **var** this\_x = xVector[j];
765. **var** func\_res = U(this\_x, this\_t);
766. **if** (j == 0) {
767. this\_max = func\_res;
768. this\_min = func\_res;
769. }
770. **else** {
771. this\_max = (func\_res > this\_max ? func\_res : this\_max);
772. this\_min = (func\_res < this\_min ? func\_res : this\_min);
773. }
774. timeVector.push(func\_res);
775. }
776. etalon.push(timeVector);
777. maxVals.push(this\_max);
778. minVals.push(this\_min);
779. **if** (i == 0) {
780. maxMaxVal = this\_max;
781. minMinVal = this\_min;
782. }
783. **else** {
784. maxMaxVal = (this\_max > maxMaxVal ? this\_max : maxMaxVal);
785. minMinVal = (this\_min < minMinVal ? this\_min : minMinVal);
786. }
787. }
788. *//задаем начальное условие*
789. **var** psiGrid0 = [];
790. **var** emptV = [];
791. **for** (**var** j = 0 ; j <= N ; ++j) {
792. psiGrid0.push(psi(xVector[j]));
793. emptV.push(0);
794. }
795. solve.push(psiGrid0);
797. **var** vector0 = psiGrid0;
798. **var** vector1 = copyV(emptV);
800. *//scheme 1*
801. **if** (scheme\_type == 1) {
803. **for** (**var** i = 1 ; i <= K ; ++i) {
805. **var** resV = explicit\_step(i, vector0, vector1);
807. solve.push(copyV(resV));
809. vector0 = copyV(resV);
810. vector1 = copyV(emptV);
812. }
814. }
815. **else** **if** (scheme\_type == 2) {
817. **for** (**var** i = 1 ; i <= K ; ++i) {
819. **var** resV = implicit\_step(i, vector0, vector1);
821. solve.push(copyV(resV));
823. vector0 = copyV(resV);
824. vector1 = copyV(emptV);
826. }
827. }
829. **else** **if** (scheme\_type == 3) {
831. **for** (**var** i = 1 ; i <= K ; ++i) {
833. **var** resV1 = explicit\_step(i, vector0, vector1);
834. **var** resV2 = implicit\_step(i, vector0, vector1);
836. **var** totalV = [];
838. **for** (**var** y = 0 ; y < resV1.length ; ++y) {
839. **var** resT = 0.5\*resV1[y] + 0.5\*resV2[y];
840. totalV.push(resT);
841. }
843. solve.push(copyV(totalV));
845. vector0 = copyV(totalV);
846. vector1 = copyV(emptV);
848. }
849. }

852. *// error*

855. absolute\_error = [];
856. relative\_error = [];
857. maxMaxValE1 = 1;
858. minMinValE1 = 0;
859. maxMaxValE2 = 1;
860. minMinValE2 = 0;
862. maxValsE1 = [];
863. minValsE1 = [];
864. maxValsE2 = [];
865. minValsE2 = [];
867. EotT = [];
869. **for** (**var** i = 0 ; i <= K ; ++i) {
870. **var** timeVector1 = [];
871. **var** timeVector2 = [];
872. **var** this\_t = t \* i;
873. **var** this\_max1 = 0;
874. **var** this\_min1 = 0;
876. **var** this\_max2 = 0;
877. **var** this\_min2 = 0;
879. **for** (**var** j = 0 ; j <= N ; ++j) {
880. **var** this\_x = xVector[j];
882. **var** err1 = solve[i][j] - etalon[i][j];
883. **var** err2 = 0;
884. **if** (Math.abs(etalon[i][j]) > 0.1) {
885. err2 = Math.abs(1 - solve[i][j]/etalon[i][j]);
886. }
887. **else** {
888. err2 = Math.abs(1 - (solve[i][j]+0.2)/(etalon[i][j] + 0.2));
889. }
891. **if** (j == 0) {
892. this\_max1 = err1;
893. this\_min1 = err1;
895. this\_max2 = err2;
896. this\_min2 = err2;
897. }
898. **else** {
899. this\_max1 = (err1 > this\_max1 ? err1 : this\_max1);
900. this\_min1 = (err1 < this\_min1 ? err1 : this\_min1);
902. this\_max2 = (err2 > this\_max2 ? err2 : this\_max2);
903. this\_min2 = (err2 < this\_min2 ? err2 : this\_min2);
904. }
905. timeVector1.push(err1);
906. timeVector2.push(err2);
907. }
909. absolute\_error.push(timeVector1);
910. relative\_error.push(timeVector2);
912. maxValsE1.push(this\_max1);
913. minValsE1.push(this\_min1);
915. maxValsE2.push(this\_max2);
916. minValsE2.push(this\_min2);

919. **if** (i == 0) {
920. maxMaxValE1 = this\_max1;
921. minMinValE1 = this\_min1;
923. maxMaxValE2 = this\_max2;
924. minMinValE2 = this\_min2;
926. **var** stepEot = Math.max(maxMaxValE1, Math.abs(minMinValE1));
928. EotT.push(stepEot);
930. minEotT = stepEot;
931. maxEotT = stepEot;
932. }
933. **else** {
934. maxMaxValE1 = (this\_max1 > maxMaxValE1 ? this\_max1 : maxMaxValE1);
935. minMinValE1 = (this\_min1 < minMinValE1 ? this\_min1 : minMinValE1);
937. **var** stepEot = Math.max(this\_max1, Math.abs(this\_min1));
939. minEotT = (stepEot < minEotT ? stepEot : minEotT);
940. maxEotT = (stepEot > maxEotT ? stepEot : maxEotT);
942. EotT.push(stepEot);


946. maxMaxValE2 = (this\_max2 > maxMaxValE2 ? this\_max2 : maxMaxValE2);
947. minMinValE2 = (this\_min2 < minMinValE2 ? this\_min2 : minMinValE2);
948. }
949. }


953. makeEtalonGraph();
954. makeSolveGraph();
955. makeErrorGraph();
956. makeErrorTGraph();
958. go = **true**;
959. }
961. **function** transfX(x) {
962. **var** newX = 60 + (567 \* (x / l));
963. **return** newX;
964. }
966. **function** transfY(y) {
967. **var** newY = 0;
968. **if** (maxVals.length > 0 && scale\_type == 2) {
969. newY = 418.6 - (382.6 \* (y - minVals[s\_cur])/(maxVals[s\_cur] - minVals[s\_cur]));
970. }
971. **else** {
972. newY = 418.6 - (382.6 \* (y - minMinVal)/(maxMaxVal - minMinVal));
973. }
974. **return** newY;
975. }

978. **function** transfEX(x) {
979. **var** newX = 60.6 + (344.2 \* (x / l));
980. **return** newX;
981. }
983. **function** transfE2X(x) {
984. **var** newX = 60.6 + (344.2 \* (x / T));
985. **return** newX;
986. }
988. **function** transfEY(y) {
989. **var** newY = 0;
990. **if** (maxValsE.length > 0 && scale\_type == 2) {
991. newY = 700 - (231 \* (y - minValsE[s\_cur])/(maxValsE[s\_cur] - minValsE[s\_cur]));
992. }
993. **else** {
994. newY = 700 - (231 \* (y - minMinValE)/(maxMaxValE - minMinValE));
995. }
996. **return** newY;
997. }
999. **function** transfE2Y(y) {
1000. **var** newY = 0;
1001. newY = 700 - (231 \* (y - minEotT)/(maxEotT - minEotT));
1003. **return** newY;
1004. }

1007. **function** makeEtalonGraph() {
1008. **for** (**var** i = 0 ; i < N ; ++i) {
1009. **var** join = **new** lib.line();
1010. stage.addChild(join);
1011. join.x = transfX(xVector[i]);
1012. join.y = transfY(etalon[0][i]);
1013. join.endX = transfX(xVector[i+1]);
1014. join.endY = transfY(etalon[0][i+1]);
1016. join.gotoAndStop(0);
1017. join.num = i;
1019. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1021. join.scaleX = join.len;
1022. join.scaleY = 1;
1024. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1026. join.gen = generation;

1029. join.visible = **true**;
1030. join.alpha = 1;
1032. join.addEventListener('tick', setPoses);
1033. }
1035. }
1037. **function** makeSolveGraph() {
1038. **for** (**var** i = 0 ; i < N ; ++i) {
1039. **var** join = **new** lib.line();
1040. stage.addChild(join);
1041. join.x = transfX(xVector[i]);
1042. join.y = transfY(solve[0][i]);
1043. join.endX = transfX(xVector[i+1]);
1044. join.endY = transfY(solve[0][i+1]);
1046. join.gotoAndStop(1);
1047. join.num = i;
1049. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1051. join.scaleX = join.len;
1052. join.scaleY = 1;
1054. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1056. join.gen = generation;

1059. join.visible = **true**;
1060. join.alpha = 1;
1062. join.addEventListener('tick', setPoses2);
1063. }
1065. }
1067. **function** makeErrorGraph() {
1068. setError();
1069. **for** (**var** i = 0 ; i < N ; ++i) {
1070. **var** join = **new** lib.line();
1071. stage.addChild(join);
1072. join.x = transfEX(xVector[i]);
1073. join.y = transfEY(errorGraph[0][i]);
1074. join.endX = transfEX(xVector[i+1]);
1075. join.endY = transfEY(errorGraph[0][i+1]);
1077. join.gotoAndStop(3);
1078. join.num = i;
1080. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1082. join.scaleX = join.len;
1083. join.scaleY = 1;
1085. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1087. join.gen = generation;

1090. join.visible = **true**;
1091. join.alpha = 1;
1093. join.addEventListener('tick', setPoses3);
1094. }
1096. }
1098. **function** makeErrorTGraph() {
1099. **for** (**var** i = 0 ; i < K ; ++i) {
1100. **var** join = **new** lib.line();
1101. stage.addChild(join);
1102. join.x = transfE2X(tVector[i]);
1103. join.y = transfE2Y(EotT[i]);
1104. join.endX = transfE2X(tVector[i+1]);
1105. join.endY = transfE2Y(tVector[i+1]);
1107. join.gotoAndStop(3);
1108. join.num = i;
1110. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1112. join.scaleX = join.len;
1113. join.scaleY = 1;
1115. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1117. join.gen = generation;

1120. join.visible = **true**;
1121. join.alpha = 1;
1123. join.addEventListener('tick', setPoses4);
1124. }
1126. }
1128. **function** setPoses(e) {
1129. **var** object = e.currentTarget;
1130. **if** (object.gen == generation) {
1131. object.x = transfX(xVector[object.num]);
1132. object.y = transfY(etalon[s\_cur][object.num]);
1133. object.endX = transfX(xVector[object.num+1]);
1134. object.endY = transfY(etalon[s\_cur][object.num+1]);
1136. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
1138. object.scaleX = object.len;
1139. object.scaleY = 1;
1141. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1142. }
1143. **else** **if** (object.gen != generation) {
1144. object.alpha -= 3/30;
1145. }
1147. **if** (object.alpha <= 0) {
1148. object.alpha = 0;
1149. object.visible = **false**;
1150. object.removeEventListener('tick', setPoses);
1151. stage.removeChild(object);
1152. }
1154. }

1157. **function** setPoses2(e) {
1158. **var** object = e.currentTarget;
1159. **if** (object.gen == generation) {
1160. object.x = transfX(xVector[object.num]);
1161. object.y = transfY(solve[s\_cur][object.num]);
1162. object.endX = transfX(xVector[object.num+1]);
1163. object.endY = transfY(solve[s\_cur][object.num+1]);
1165. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
1167. object.scaleX = object.len;
1168. object.scaleY = 1;
1170. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1171. }
1172. **else** **if** (object.gen != generation) {
1173. object.alpha -= 3/30;
1174. }
1176. **if** (object.alpha <= 0) {
1177. object.alpha = 0;
1178. object.visible = **false**;
1179. object.removeEventListener('tick', setPoses2);
1180. stage.removeChild(object);
1181. }
1183. }

1186. **function** setPoses3(e) {
1187. **var** object = e.currentTarget;
1188. **if** (object.gen == generation) {
1189. object.x = transfEX(xVector[object.num]);
1190. object.y = transfEY(errorGraph[s\_cur][object.num]);
1191. object.endX = transfEX(xVector[object.num+1]);
1192. object.endY = transfEY(errorGraph[s\_cur][object.num+1]);
1194. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
1196. object.scaleX = object.len;
1197. object.scaleY = 1;
1199. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1200. }
1201. **else** **if** (object.gen != generation) {
1202. object.alpha -= 3/30;
1203. }
1205. **if** (pogr\_type <= 2) {
1206. object.visible = **true**;
1207. }
1208. **else** {
1209. object.visible = **false**;
1210. }
1212. **if** (object.alpha <= 0) {
1213. object.alpha = 0;
1214. object.visible = **false**;
1215. object.removeEventListener('tick', setPoses3);
1216. stage.removeChild(object);
1217. }
1219. }

1222. **function** setPoses4(e) {
1223. **var** object = e.currentTarget;
1224. **if** (object.gen == generation) {
1225. object.x = transfE2X(tVector[object.num]);
1226. object.y = transfE2Y(EotT[object.num]);
1227. object.endX = transfE2X(tVector[object.num+1]);
1228. object.endY = transfE2Y(EotT[object.num+1]);
1230. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
1232. object.scaleX = object.len;
1233. object.scaleY = 1;
1235. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1236. }
1237. **else** **if** (object.gen != generation) {
1238. object.alpha -= 3/30;
1239. }
1241. **if** (pogr\_type == 3) {
1242. object.visible = **true**;
1243. }
1244. **else** {
1245. object.visible = **false**;
1246. }
1248. **if** (object.alpha <= 0) {
1249. object.alpha = 0;
1250. object.visible = **false**;
1251. object.removeEventListener('tick', setPoses4);
1252. stage.removeChild(object);
1253. }
1255. }







1264. *//метод прогонки*
1265. **function** progonka(matrix, vectorB) {
1266. **var** vectorX = [];
1268. **var** N = vectorB.length;
1270. **var** alphas = [];
1271. **var** betas = [];

1274. **for** (**var** i = 0 ; i < vectorB.length ; ++i) {
1275. alphas.push(0);
1276. betas.push(0);
1277. }
1279. *//Прямой ход прогонки*
1281. **for** (**var** i = 0; i < N; ++i) {
1282. **var** A0, C0, B0, F0;
1284. **if** (i - 1 < 0) {
1285. A0 = 0;
1286. }
1287. **else** {
1288. A0 = matrix[i][i - 1];
1289. }
1291. C0 = -1 \* matrix[i][i];
1293. **if** (i + 1 < N) {
1294. B0 = matrix[i][i + 1];
1295. }
1296. **else** {
1297. B0 = 0;
1298. }
1300. F0 = vectorB[i];

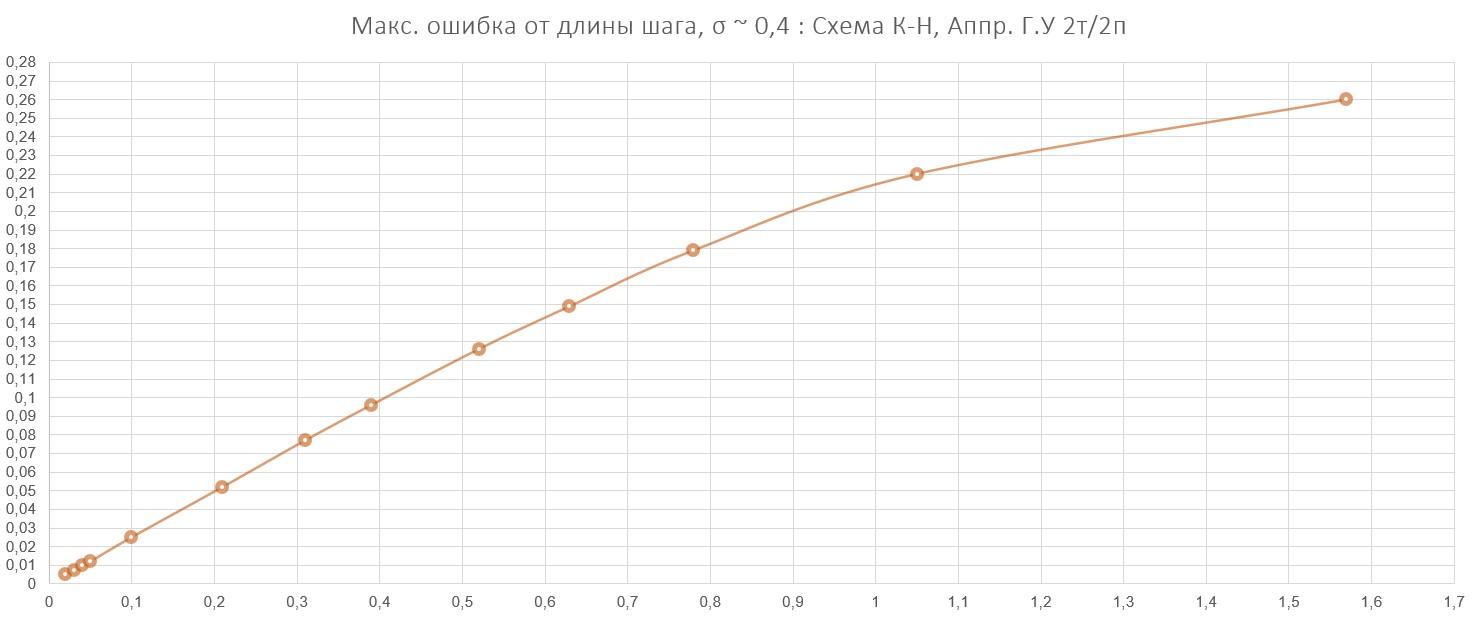
1303. **if** (i == 0) {
1304. alphas[i] = B0 / C0;
1305. betas[i] = -(F0 / C0);
1306. }
1307. **else** **if** (i == N - 1) {
1308. alphas[i] = 0;
1309. betas[i] = (betas[i - 1] \* A0 - F0) / (C0 - alphas[i - 1] \* A0);
1310. }
1311. **else** {
1312. alphas[i] = B0 / (C0 - alphas[i - 1] \* A0);
1313. betas[i] = (betas[i - 1] \* A0 - F0) / (C0 - alphas[i - 1] \* A0);
1314. }
1316. }

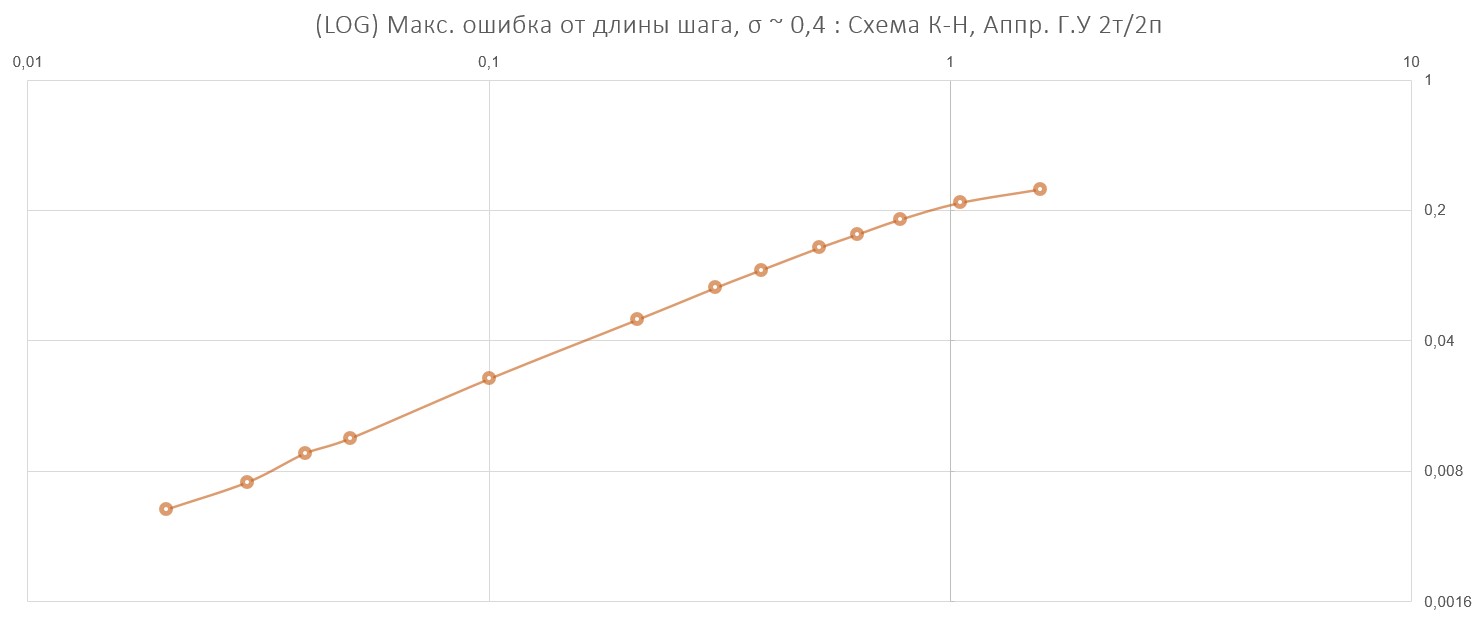
1319. vectorX[N - 1] = betas[N - 1];
1321. **for** (**var** i = 2; i <= N; ++i) {
1322. vectorX[N - i] = alphas[N - i] \* vectorX[N - i + 1] + betas[N - i];
1323. }
1325. **return** vectorX;
1326. }
1327. Графики

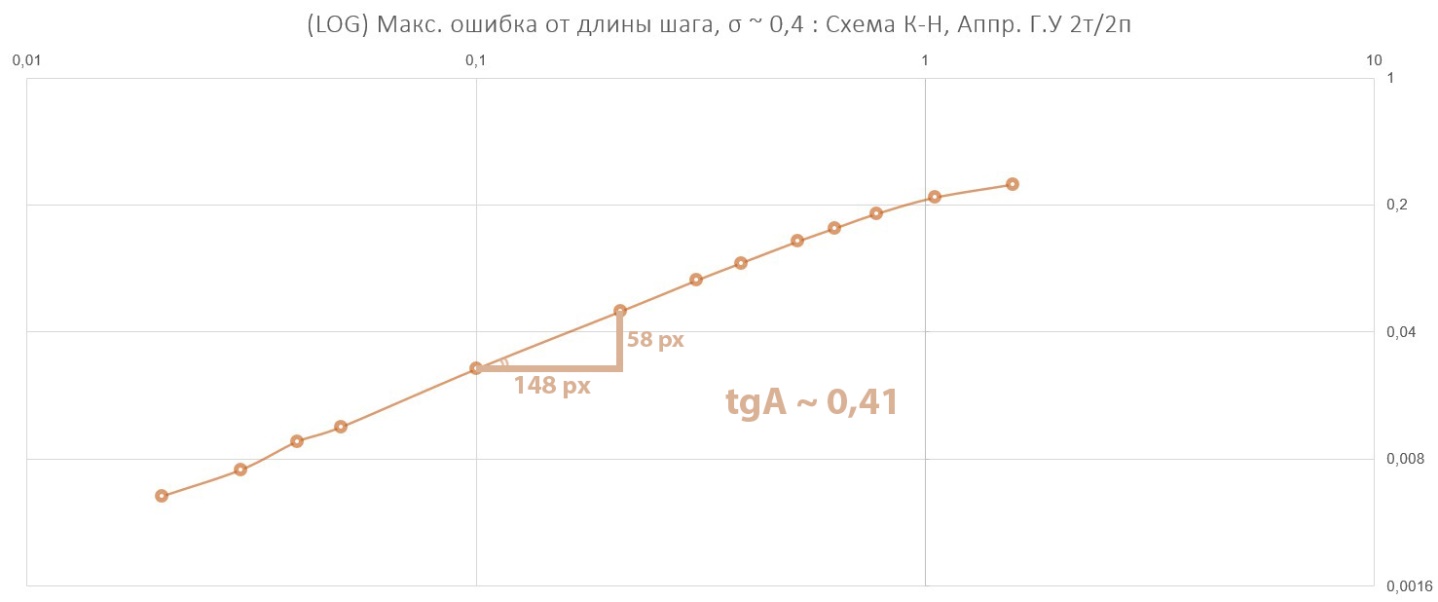
Ошибка от времени при T = 10



Ошибка от размера шага – обычный и логарифмические графики







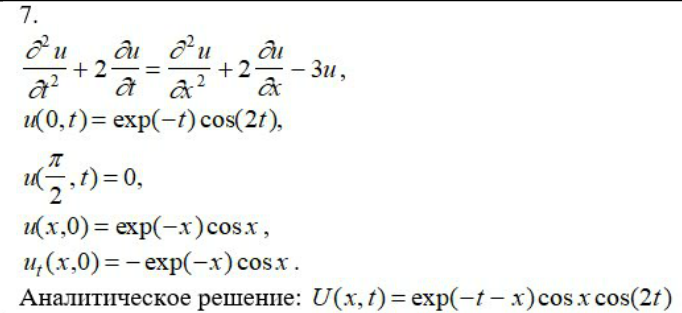
1. Данная лабораторная работа выполнена: **17 января 2021.**

**Лабораторная работа 6**

1. Тема ЛР:

Метод конечных разностей для решения уравнений гиперболического типа. Явная крест-схема и неявная.

1. Вариант : **7**



1. Алгоритм:

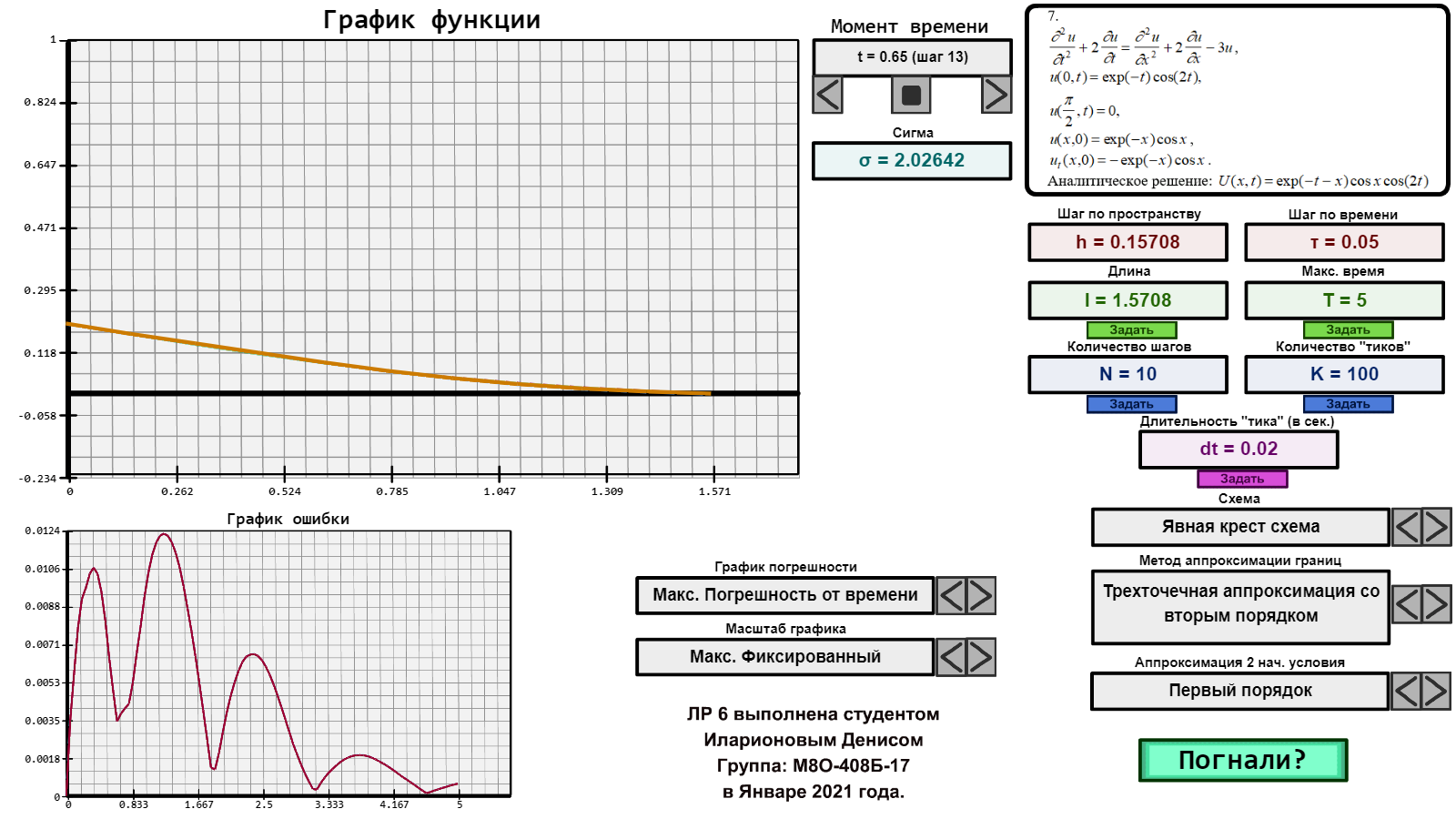
После предыдущей лабораторной, эта мне показалась куда легче. Используются похожие схемы, однако, теперь, я знаю, как решать такие уравнения. Явная крест схема использует значения двух предыдущих временных слоев. Поэтому, здесь было кое-что необычное – аппроксимация второго начального условия. Именно так мы находим значения во втором временном слое. К счастью, в моем варианте границы были заданы явно, так что, не пришлось долго возиться с этим.

1. Среда разработки:

Adobe Animate, Javascript + HTML5

1. Реализация

Можно указать порядок аппроксимации 2 начального условия. Выбрать схему. Посмотреть погрешность, а также все то, что было в предыдущей лабораторной.



1. Код программы
2. **var** alpha = 0; *//α*
3. **var** beta = 1; *//β*
4. **var** gamma = 0; *//γ*
5. **var** delta = 1; *//δ*

8. *//α \* Ux(0, t) + β \* U(0, t) = exp(-t)cos(2t)*
9. *//γ \* Ux(l, t) + δ \* U(l, t) = 0*
10. *//Обозначим все как граничные условия 3 рода*
12. **function** phi\_0(t) {
13. **return** Math.exp(-1 \* t) \* Math.cos(2\*t);
14. }
16. **function** phi\_l(t) {
17. **return** 0;
18. }
20. **function** psi(x) {
21. **return** Math.exp(-x) \* Math.cos(x);
22. }
23. *// Начальное условие 1 (U(x, 0))*
25. **function** psi2(x) {
26. **return** -Math.exp(-x) \* Math.cos(x);
27. }
28. *// Начальное условие 2 (Ut(x, 0))*

31. **function** psi2d1(x) {
32. **return** Math.exp(-x)\*Math.sin(x) + Math.exp(-x)\*Math.cos(x);
33. }

36. *//Сделать анализ по результатам*
37. *//Порядок сходимости метода*
38. *//Зависимость при const числе сигма (куррента)*
39. *//Макс. Погрешность от шага (в опред. момент времени)*
40. *//Лучше логарифм*
41. *//Сделать график зависимости модуля макс. ошибки от времени*



46. **function** psi2d2(x) {
47. **return** -2 \* Math.exp(-x) \* Math.sin(x);
48. }
49. *// Первая и вторая производная 2 н.у*
51. **function** U(x, t) {
52. **return** Math.exp(-t - x) \* Math.cos(x) \* Math.cos(2\*t);
53. } *//Аналитическое решение*

56. *//Параметры сетки*
57. **var** l = 3.1415926535/2; *//интервал*
58. **var** N = 10; *//количество шагов*
59. **var** T = 1; *//макс. время*
60. **var** K = 100; *//количество тиков*
62. **var** h = 0; **var** t = 0;
64. **var** a = 1; **var** b = 2; **var** c = -3;

67. **var** sigma = 1;
68. *//σ = a^2 \* τ / h^2*
70. **var** dt = 0.02; *//длительность тика*
71. **var** cur\_tick = 0;

74. **var** generation = 0; *//номер генерации*
75. **var** go = **false**; *//увеличивать счетчик тиков?*

78. **this**.addEventListener("tick", det\_ht.bind(**this**));
79. **function** det\_ht() {
80. h = l / N;
81. t = T / K;
83. sigma = a \* a \* t / (h \* h);
84. *//определяем шаги по времени и пространству*
86. **this**.h\_text.text = "h = " + Math.round(h \* 100000)/100000;
87. **this**.tau\_text.text = "τ = " + Math.round(t \* 100000)/100000;
89. **this**.l\_text.text = "l = " + Math.round(l \* 100000)/100000;
90. **this**.T\_text.text = "T = " + Math.round(T \* 100000)/100000;
92. **this**.N\_text.text = "N = " + Math.round(N);
93. **this**.K\_text.text = "K = " + Math.round(K);
95. **this**.dt\_text.text = "dt = " + Math.round(dt\*1000)/1000;
97. **this**.sig\_text.text = "σ = " + Math.round(sigma \* 100000)/100000;

100. **if** (go) {
101. cur\_tick += 1/60;
102. **if** (cur\_tick >= dt && s\_cur < K) {
103. **var** tGot = Math.floor(cur\_tick / dt);
104. cur\_tick %= dt;
105. s\_cur += Math.min(K - s\_cur, tGot);
106. }
107. **else** **if** (s\_cur == K) {
108. go = **false**;
109. }
110. }
112. }



117. **this**.setGreen1.addEventListener("click", set\_l\_prompt.bind(**this**));
118. **function** set\_l\_prompt() {
119. generation += 1;
120. **var** temp = prompt("Введите l:", '');
121. temp = Number.parseFloat(temp);
122. **if** (isNaN(temp)) {
123. *//если ввели какую-то хрень*
124. temp = l;
125. }
126. **else** **if** (temp > 1000000) {
127. *//ограничение на величину*
128. temp = 1000000;
129. }
130. **else** **if** (temp <= 0) {
131. *//всегда положительно*
132. temp = 0.001;
133. }
134. l = temp;
135. }


139. **this**.setGreen2.addEventListener("click", set\_T\_prompt.bind(**this**));
140. **function** set\_T\_prompt() {
141. generation += 1;
142. **var** temp = prompt("Введите T:", '');
143. temp = Number.parseFloat(temp);
144. **if** (isNaN(temp)) {
145. *//если ввели какую-то хрень*
146. temp = T;
147. }
148. **else** **if** (temp > 1000000) {
149. *//ограничение на величину*
150. temp = 1000000;
151. }
152. **else** **if** (temp <= 0) {
153. *//всегда положительно*
154. temp = 0.001;
155. }
156. T = temp;
157. }

160. **this**.setBlue1.addEventListener("click", set\_N\_prompt.bind(**this**));
161. **function** set\_N\_prompt() {
162. generation += 1;
163. **var** temp = prompt("Введите N:", '');
164. temp = Number.parseInt(temp);
165. **if** (isNaN(temp)) {
166. *//если ввели какую-то хрень*
167. temp = N;
168. }
169. **else** **if** (temp > 200) {
170. *//ограничение на величину*
171. temp = 200;
172. }
173. **else** **if** (temp <= 0) {
174. *//всегда положительно*
175. temp = 1;
176. }
177. N = temp;
178. }

181. **this**.setBlue2.addEventListener("click", set\_K\_prompt.bind(**this**));
182. **function** set\_K\_prompt() {
183. generation += 1;
184. **var** temp = prompt("Введите K:", '');
185. temp = Number.parseInt(temp);
186. **if** (isNaN(temp)) {
187. *//если ввели какую-то хрень*
188. temp = K;
189. }
190. **else** **if** (temp > 10000) {
191. *//ограничение на величину*
192. temp = 10000;
193. }
194. **else** **if** (temp <= 0) {
195. *//всегда положительно*
196. temp = 1;
197. }
198. K = temp;
199. }

202. **this**.setPurp1.addEventListener("click", set\_dt\_prompt.bind(**this**));
203. **function** set\_dt\_prompt() {
204. cur\_tick = 0;
205. **var** temp = prompt("Введите dt (для ручного переключения можно ввести очень большим):", '');
206. temp = Number.parseFloat(temp);
207. **if** (isNaN(temp)) {
208. *//если ввели какую-то хрень*
209. temp = dt;
210. }
211. **else** **if** (temp > 1000000) {
212. *//ограничение на величину*
213. temp = 1000000;
214. }
215. **else** **if** (temp < 0.02) {
216. *//всегда положительно и больше 0.2 (почти плавная смена графика)*
217. temp = 0.02;
218. }
219. dt = temp;
220. }


224. **var** scheme\_type = 1;
225. *//Тип схемы*
226. *//1,2 - Явная/Неявная схема*
228. **var** theta = 1;

231. **var** meth\_type = 1;
232. *//Метод аппроксимации г.у*
233. *//1 - Двухточечная аппроксимация с первым порядком*
234. *//2 - Трехточечная аппроксимация со вторым порядком*
235. *//3 - Двухточечная аппроксимация со вторым порядком*
237. **var** approx\_type = 1;
238. *//Метод аппроксимации 2 н.у*
239. *//1 - Первый порядок*
240. *//2 - Второй порядок*


244. **var** pogr\_type = 1;
245. *//Тип погрешности*
246. *//1 - Абсолютная*
247. *//2 - Относительная*
248. *//3 - От времени*
250. **var** scale\_type = 1;
251. *//Масштаб графика*
252. *//1 - Фиксированный*
253. *//2 - Динамический*
255. **this**.SLeft1.addEventListener("click", scheme\_left.bind(**this**));
256. **function** scheme\_left() {
257. **if** (scheme\_type == 1) {
258. scheme\_type = 2;
259. }
260. **else** {
261. scheme\_type -= 1;
262. }
263. }
265. **this**.SRight1.addEventListener("click", scheme\_right.bind(**this**));
266. **function** scheme\_right() {
267. **if** (scheme\_type == 2) {
268. scheme\_type = 1;
269. }
270. **else** {
271. scheme\_type += 1;
272. }
273. }

276. **this**.SLeft2.addEventListener("click", meth\_left.bind(**this**));
277. **function** meth\_left() {
278. **if** (meth\_type == 1) {
279. meth\_type = 3;
280. }
281. **else** {
282. meth\_type -= 1;
283. }
284. }
286. **this**.SRight2.addEventListener("click", meth\_right.bind(**this**));
287. **function** meth\_right() {
288. **if** (meth\_type == 3) {
289. meth\_type = 1;
290. }
291. **else** {
292. meth\_type += 1;
293. }
294. }

297. **this**.SLeft6.addEventListener("click", approx\_left.bind(**this**));
298. **function** approx\_left() {
299. **if** (approx\_type == 1) {
300. approx\_type = 2;
301. }
302. **else** {
303. approx\_type -= 1;
304. }
305. }
307. **this**.SRight6.addEventListener("click", approx\_right.bind(**this**));
308. **function** approx\_right() {
309. **if** (approx\_type == 2) {
310. approx\_type = 1;
311. }
312. **else** {
313. approx\_type += 1;
314. }
315. }

318. **this**.SLeft3.addEventListener("click", curStep\_left.bind(**this**));
319. **function** curStep\_left() {
320. **if** (s\_cur > 0) {
321. s\_cur -= 1;
322. }
323. }
325. **this**.SRight3.addEventListener("click", curStep\_right.bind(**this**));
326. **function** curStep\_right() {
327. **if** (s\_cur < K) {
328. s\_cur += 1;
329. }
330. }

333. **this**.stopBtn.addEventListener("click", curStep\_stop.bind(**this**));
334. **function** curStep\_stop() {
335. dt = 1000000;
336. }
338. **this**.SLeft4.addEventListener("click", pogr\_left.bind(**this**));
339. **function** pogr\_left() {
340. **if** (pogr\_type == 1) {
341. pogr\_type = 3;
342. }
343. **else** {
344. pogr\_type -= 1;
345. }
346. }
348. **this**.SRight4.addEventListener("click", pogr\_right.bind(**this**));
349. **function** pogr\_right() {
350. **if** (pogr\_type == 3) {
351. pogr\_type = 1;
352. }
353. **else** {
354. pogr\_type += 1;
355. }
356. }
358. **this**.SLeft5.addEventListener("click", scale\_left.bind(**this**));
359. **function** scale\_left() {
360. **if** (scale\_type == 1) {
361. scale\_type = 2;
362. }
363. **else** {
364. scale\_type -= 1;
365. }
366. }
368. **this**.SRight5.addEventListener("click", scale\_right.bind(**this**));
369. **function** scale\_right() {
370. **if** (scale\_type == 2) {
371. scale\_type = 1;
372. }
373. **else** {
374. scale\_type += 1;
375. }
376. }
378. **this**.addEventListener("tick", setTexts2.bind(**this**));
379. **function** setTexts2() {
380. **if** (scheme\_type == 1) {
381. **this**.scheme\_text.text = "Явная крест схема";
382. theta = 1;
383. }
384. **else** **if** (scheme\_type == 2) {
385. **this**.scheme\_text.text = "Неявная схема";
386. theta = 0;
387. }
389. **if** (meth\_type == 1) {
390. **this**.meth\_text.text = "Двухточечная аппроксимация с первым порядком";
391. }
392. **else** **if** (meth\_type == 2) {
393. **this**.meth\_text.text = "Трехточечная аппроксимация со вторым порядком";
394. }
395. **else** **if** (meth\_type == 3) {
396. **this**.meth\_text.text = "Двухточечная аппроксимация со вторым порядком";
397. }
399. **if** (approx\_type == 1) {
400. **this**.approx\_text.text = "Первый порядок";
401. theta = 1;
402. }
403. **else** **if** (approx\_type == 2) {
404. **this**.approx\_text.text = "Второй порядок";
405. theta = 0;
406. }

409. **if** (pogr\_type == 1) {
410. **this**.pogr\_text.text = "Абсолютная погрешность";
411. }
412. **else** **if** (pogr\_type == 2) {
413. **this**.pogr\_text.text = "Относительная погрешность";
414. }
415. **else** **if** (pogr\_type == 3) {
416. **this**.pogr\_text.text = "Макс. Погрешность от времени";
417. }

420. **if** (scale\_type == 1) {
421. **this**.scale\_text.text = "Макс. Фиксированный";
422. }
423. **else** **if** (scale\_type == 2) {
424. **this**.scale\_text.text = "Динамический";
425. }

428. **this**.divDown0.text = "" + 0;
429. **this**.divDown1.text = "" + Math.round(l\*1/6\*1000)/1000;
430. **this**.divDown2.text = "" + Math.round(l\*2/6\*1000)/1000;
431. **this**.divDown3.text = "" + Math.round(l\*3/6\*1000)/1000;
432. **this**.divDown4.text = "" + Math.round(l\*4/6\*1000)/1000;
433. **this**.divDown5.text = "" + Math.round(l\*5/6\*1000)/1000;
434. **this**.divDown6.text = "" + Math.round(l\*1000)/1000;

437. **if** (pogr\_type <= 2) {
438. **this**.divDown0err.text = "" + 0;
439. **this**.divDown1err.text = "" + Math.round(l\*1/6\*1000)/1000;
440. **this**.divDown2err.text = "" + Math.round(l\*2/6\*1000)/1000;
441. **this**.divDown3err.text = "" + Math.round(l\*3/6\*1000)/1000;
442. **this**.divDown4err.text = "" + Math.round(l\*4/6\*1000)/1000;
443. **this**.divDown5err.text = "" + Math.round(l\*5/6\*1000)/1000;
444. **this**.divDown6err.text = "" + Math.round(l\*1000)/1000;
445. }
446. **else** {
447. **this**.divDown0err.text = "" + 0;
448. **this**.divDown1err.text = "" + Math.round(T\*1/6\*1000)/1000;
449. **this**.divDown2err.text = "" + Math.round(T\*2/6\*1000)/1000;
450. **this**.divDown3err.text = "" + Math.round(T\*3/6\*1000)/1000;
451. **this**.divDown4err.text = "" + Math.round(T\*4/6\*1000)/1000;
452. **this**.divDown5err.text = "" + Math.round(T\*5/6\*1000)/1000;
453. **this**.divDown6err.text = "" + Math.round(T\*1000)/1000;
454. }

457. **if** (maxVals.length > 0 && scale\_type == 2) {
458. **this**.divUp0.text = "" + Math.round(minVals[s\_cur]\*1000)/1000;
459. **this**.divUp1.text = "" + Math.round((minVals[s\_cur] + (maxVals[s\_cur] - minVals[s\_cur])\*1/7)\*1000)/1000;
460. **this**.divUp2.text = "" + Math.round((minVals[s\_cur] + (maxVals[s\_cur] - minVals[s\_cur])\*2/7)\*1000)/1000;
461. **this**.divUp3.text = "" + Math.round((minVals[s\_cur] + (maxVals[s\_cur] - minVals[s\_cur])\*3/7)\*1000)/1000;
462. **this**.divUp4.text = "" + Math.round((minVals[s\_cur] + (maxVals[s\_cur] - minVals[s\_cur])\*4/7)\*1000)/1000;
463. **this**.divUp5.text = "" + Math.round((minVals[s\_cur] + (maxVals[s\_cur] - minVals[s\_cur])\*5/7)\*1000)/1000;
464. **this**.divUp6.text = "" + Math.round((minVals[s\_cur] + (maxVals[s\_cur] - minVals[s\_cur])\*6/7)\*1000)/1000;
465. **this**.divUp7.text = "" + Math.round(maxVals[s\_cur]\*1000)/1000;
466. }
467. **else** {
468. **this**.divUp0.text = "" + Math.round(minMinVal\*1000)/1000;
469. **this**.divUp1.text = "" + Math.round((minMinVal + (maxMaxVal - minMinVal)\*1/7)\*1000)/1000;
470. **this**.divUp2.text = "" + Math.round((minMinVal + (maxMaxVal - minMinVal)\*2/7)\*1000)/1000;
471. **this**.divUp3.text = "" + Math.round((minMinVal + (maxMaxVal - minMinVal)\*3/7)\*1000)/1000;
472. **this**.divUp4.text = "" + Math.round((minMinVal + (maxMaxVal - minMinVal)\*4/7)\*1000)/1000;
473. **this**.divUp5.text = "" + Math.round((minMinVal + (maxMaxVal - minMinVal)\*5/7)\*1000)/1000;
474. **this**.divUp6.text = "" + Math.round((minMinVal + (maxMaxVal - minMinVal)\*6/7)\*1000)/1000;
475. **this**.divUp7.text = "" + Math.round(maxMaxVal\*1000)/1000;
476. }
478. **if** (maxValsE.length > 0 && scale\_type == 2 && pogr\_type <= 2) {
479. **this**.divUp0err.text = "" + Math.round(minValsE[s\_cur]\*1000)/1000;
480. **this**.divUp1err.text = "" + Math.round((minValsE[s\_cur] + (maxValsE[s\_cur] - minValsE[s\_cur])\*1/7)\*1000)/1000;
481. **this**.divUp2err.text = "" + Math.round((minValsE[s\_cur] + (maxValsE[s\_cur] - minValsE[s\_cur])\*2/7)\*1000)/1000;
482. **this**.divUp3err.text = "" + Math.round((minValsE[s\_cur] + (maxValsE[s\_cur] - minValsE[s\_cur])\*3/7)\*1000)/1000;
483. **this**.divUp4err.text = "" + Math.round((minValsE[s\_cur] + (maxValsE[s\_cur] - minValsE[s\_cur])\*4/7)\*1000)/1000;
484. **this**.divUp5err.text = "" + Math.round((minValsE[s\_cur] + (maxValsE[s\_cur] - minValsE[s\_cur])\*5/7)\*1000)/1000;
485. **this**.divUp6err.text = "" + Math.round((minValsE[s\_cur] + (maxValsE[s\_cur] - minValsE[s\_cur])\*6/7)\*1000)/1000;
486. **this**.divUp7err.text = "" + Math.round(maxValsE[s\_cur]\*1000)/1000;
487. }
488. **else** **if** (pogr\_type <= 2) {
489. **this**.divUp0err.text = "" + Math.round(minMinValE\*1000)/1000;
490. **this**.divUp1err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*1/7)\*1000)/1000;
491. **this**.divUp2err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*2/7)\*1000)/1000;
492. **this**.divUp3err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*3/7)\*1000)/1000;
493. **this**.divUp4err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*4/7)\*1000)/1000;
494. **this**.divUp5err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*5/7)\*1000)/1000;
495. **this**.divUp6err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*6/7)\*1000)/1000;
496. **this**.divUp7err.text = "" + Math.round(maxMaxValE\*1000)/1000;
497. }
498. **else** {
499. **this**.divUp0err.text = "" + Math.round(minEotT\*10000)/10000;
500. **this**.divUp1err.text = "" + Math.round((minEotT + (maxEotT - minEotT)\*1/7)\*10000)/10000;
501. **this**.divUp2err.text = "" + Math.round((minEotT + (maxEotT - minEotT)\*2/7)\*10000)/10000;
502. **this**.divUp3err.text = "" + Math.round((minEotT + (maxEotT - minEotT)\*3/7)\*10000)/10000;
503. **this**.divUp4err.text = "" + Math.round((minEotT + (maxEotT - minEotT)\*4/7)\*10000)/10000;
504. **this**.divUp5err.text = "" + Math.round((minEotT + (maxEotT - minEotT)\*5/7)\*10000)/10000;
505. **this**.divUp6err.text = "" + Math.round((minEotT + (maxEotT - minEotT)\*6/7)\*10000)/10000;
506. **this**.divUp7err.text = "" + Math.round(maxEotT\*10000)/10000;
507. }
509. t\_cur = s\_cur\*t;
511. **this**.step\_text.text = "t = " + Math.round(t\_cur\*10000)/10000 + " (шаг " + Math.round(s\_cur) + ")";
513. **this**.axisDown.y = Math.min(Math.max(transfY(0), 26.4), 418.6);
515. **this**.axisDownErr.y = Math.min(Math.max(transfEY(0), 468), 700);
516. }

519. *//baseGraph*
520. *//x = 0 : posX = 60*
521. *//y = 0 : posY = 418.6*
523. *//x = max : posX = 627*
524. *//y = max : posY = 26.4*
526. *//errGraph*
527. *//x = 0 : posX = 53.6*
528. *//y = 0 : posY = 700*
530. *//x = max : posX = 397.8*
531. *//y = max : posY = 468.4*




537. **var** t\_cur = 0;
538. *//текущий момент времени*
540. **var** s\_cur = 0;
541. *//текущий шаг*

544. **var** etalon = [];
545. **var** xVector = [];
546. **var** tVector = [];
548. **var** maxMaxVal = 1;
549. **var** minMinVal = 0;
551. **var** maxMaxValE = 1; *//link*
552. **var** minMinValE = 0; *//link*
553. **var** maxMaxValE1 = 1;
554. **var** minMinValE1 = 0;
555. **var** maxMaxValE2 = 1;
556. **var** minMinValE2 = 0;
558. **var** maxVals = [];
559. **var** minVals = [];
561. **var** maxValsE = []; *//link*
562. **var** minValsE = []; *//link*
563. **var** maxValsE1 = [];
564. **var** minValsE1 = [];
565. **var** maxValsE2 = [];
566. **var** minValsE2 = [];


570. **var** EotT = [];
572. **this**.addEventListener("tick", setError.bind(**this**));
573. **function** setError() {
574. **if** (pogr\_type == 1) {
575. maxMaxValE = maxMaxValE1;
576. minMinValE = minMinValE1;
577. maxValsE = maxValsE1;
578. minValsE = minValsE1;
579. errorGraph = absolute\_error;
580. }
581. **else** {
582. maxMaxValE = maxMaxValE2;
583. minMinValE = minMinValE2;
584. maxValsE = maxValsE2;
585. minValsE = minValsE2;
586. errorGraph = relative\_error;
587. }
588. }
590. **var** solve = [];

593. **function** copyV(vector) {
594. **var** newV = [];
595. **for** (**var** i = 0 ; i < vector.length ; ++i) {
596. newV.push(vector[i]);
597. }
598. **return** newV;
599. }

602. **var** absolute\_error = [];
603. **var** relative\_error = [];
604. **var** errorGraph = [];
606. **var** minEotT = 0;
607. **var** maxEotT = 1;


611. **function** explicit\_step(i, vector0, vector1, vector2) {
612. **var** curT = i \* t;
614. *//Сначала вычисляем серединку*
615. **for** (**var** j = 1 ; j < N ; ++j) {
616. *//данную формулу я долго выводил на бумаге, аппроксимируя*
617. *//производные*
619. vector2[j] = (-2\*vector1[j]\*(t\*t - h\*h - t\*h\*h - t\*t\*h
620. + 1.5\*t\*t\*h\*h) - vector0[j]\*h\*h + vector1[j+1]\*t\*t
621. - vector1[j-1]\*t\*t\*(2\*h-1)) / (h\*h\*(1 + 2\*t));



626. maxVals[i] = (vector1[j] > maxVals[i] ? vector2[j] : maxVals[i]);
627. minVals[i] = (vector1[j] < minVals[i] ? vector2[j] : minVals[i]);
628. }
630. *//аппроксимируем граничные условия*
631. *//Граничные условия 1 рода аппроксимируются с беск. порядком*
632. *//Поэтому, выбор метода аппроксимации гр. усл. не меняет ничего*

635. vector2[0] = phi\_0(curT)/beta;
636. vector2[N] = 0;

639. maxVals[i] = (vector2[0] > maxVals[i] ? vector2[0] : maxVals[i]);
640. maxVals[i] = (vector2[N] > maxVals[i] ? vector2[N] : maxVals[i]);
642. minVals[i] = (vector2[0] < minVals[i] ? vector2[0] : minVals[i]);
643. minVals[i] = (vector2[N] < minVals[i] ? vector2[N] : minVals[i]);
645. maxMaxVal = (maxVals[i] > maxMaxVal ? maxVals[i] : maxMaxVal);
646. minMinVal = (minVals[i] < minMinVal ? minVals[i] : minMinVal);
648. **return** copyV(vector2);
650. }


654. **function** implicit\_step(i, vector0, vector1, vector2) {
655. **var** curT = i \* t;
657. **var** solveMatrix = [];
658. **var** solveVector = [];
660. **var** emptV = [];
661. **for** (**var** j = 0 ; j <= N ; ++j) {
662. emptV.push(0);
663. }
665. **var** thisV = copyV(emptV);
667. thisV[0] = beta
668. thisV[1] = 0;
670. solveMatrix.push(copyV(thisV));
672. **var** d0 = phi\_0(curT);
674. solveVector.push(d0); *//d0*

677. **for** (**var** j = 1 ; j < N ; ++j) {
678. **var** thisV = copyV(emptV);
679. thisV[j - 1] = -1/(h\*h);
680. thisV[j] = 1/(t\*t) + 2/t + 2/(h\*h) + 2/h + 3;
681. thisV[j + 1] = -1/(h\*h) - 2/h;
683. solveMatrix.push(copyV(thisV));
685. **var** dj = (2\*vector1[j] - vector0[j])/(t\*t) + 2\*vector1[j]/t;
686. solveVector.push(dj);

689. }
691. **var** thisV = copyV(emptV);

694. thisV[N-1] = 0;
695. thisV[N] = delta;
697. solveMatrix.push(copyV(thisV));
699. **var** dn = 0;
701. solveVector.push(dn); *//d1*

704. **var** the\_vector = progonka(solveMatrix, solveVector);

707. **for** (**var** j = 0 ; j <= N ; ++j) {
709. maxVals[i] = (the\_vector[j] > maxVals[i] ? the\_vector[j] : maxVals[i]);
710. minVals[i] = (the\_vector[j] < minVals[i] ? the\_vector[j] : minVals[i]);
712. maxMaxVal = (maxVals[i] > maxMaxVal ? maxVals[i] : maxMaxVal);
713. minMinVal = (minVals[i] < minMinVal ? minVals[i] : minMinVal);
715. }
717. **return** copyV(the\_vector);
718. }


722. **this**.beginBtn.addEventListener("click", beginSimulation.bind(**this**));
723. **function** beginSimulation() {
724. generation += 1;
725. s\_cur = 0;
726. etalon = [];
727. solve = [];
728. maxMaxVal = 1;
729. minMinVal = 0;
731. maxVals = [];
732. minVals = [];
734. xVector = [];
735. tVector = [];
737. **for** (**var** j = 0 ; j <= N ; ++j) {
738. xVector.push(h \* j);
739. }
740. **for** (**var** i = 0 ; i <= K ; ++i) {
741. **var** timeVector = [];
742. **var** this\_t = t \* i;
743. tVector.push(this\_t);
744. **var** this\_max = 0;
745. **var** this\_min = 0;
746. **for** (**var** j = 0 ; j <= N ; ++j) {
747. **var** this\_x = xVector[j];
748. **var** func\_res = U(this\_x, this\_t);
749. **if** (j == 0) {
750. this\_max = func\_res;
751. this\_min = func\_res;
752. }
753. **else** {
754. this\_max = (func\_res > this\_max ? func\_res : this\_max);
755. this\_min = (func\_res < this\_min ? func\_res : this\_min);
756. }
757. timeVector.push(func\_res);
758. }
759. etalon.push(timeVector);
760. maxVals.push(this\_max);
761. minVals.push(this\_min);
762. **if** (i == 0) {
763. maxMaxVal = this\_max;
764. minMinVal = this\_min;
765. }
766. **else** {
767. maxMaxVal = (this\_max > maxMaxVal ? this\_max : maxMaxVal);
768. minMinVal = (this\_min < minMinVal ? this\_min : minMinVal);
769. }
770. }
772. *//задаем начальное условие*
773. **var** psiGrid0 = [];
774. **var** emptV = [];
775. **for** (**var** j = 0 ; j <= N ; ++j) {
776. psiGrid0.push(psi(xVector[j]));
777. emptV.push(0);
778. }
779. solve.push(psiGrid0);
781. **var** psiGrid1 = [];
782. *//1 порядок аппрокс. 2 нач условия*
783. **if** (approx\_type == 1) {
784. **for** (**var** j = 0 ; j <= N ; ++j) {
785. **var** ans = psi(xVector[j]) + psi2(xVector[j])\*t;
786. psiGrid1.push(ans);
787. }
788. solve.push(psiGrid1);
789. }
790. **else** **if** (approx\_type == 2) {
791. **for** (**var** j = 0 ; j <= N ; ++j) {
792. **var** ans = psi(xVector[j]) + psi2(xVector[j])\*t + (a\*a\*psi2d2(xVector[j]) + b\*psi2d1(xVector[j]) + c\*psi2(xVector[j]))\*(t\*t/2);
793. psiGrid1.push(ans);
794. }
795. solve.push(psiGrid1);
796. }



801. **var** vector0 = psiGrid0;
802. **var** vector1 = psiGrid1;
803. **var** vector2 = copyV(emptV);
805. *//scheme 1*
806. **if** (scheme\_type == 1) {
808. **for** (**var** i = 2 ; i <= K ; ++i) {
810. **var** resV = explicit\_step(i, vector0, vector1, vector2);
812. solve.push(copyV(resV));
814. vector0 = copyV(vector1);
815. vector1 = copyV(resV);
816. vector2 = copyV(emptV);
818. }
820. }
821. **else** **if** (scheme\_type == 2) {
823. **for** (**var** i = 2 ; i <= K ; ++i) {
825. **var** resV = implicit\_step(i, vector0, vector1, vector2);
827. solve.push(copyV(resV));
829. vector0 = copyV(vector1);
830. vector1 = copyV(resV);
831. vector2 = copyV(emptV);
833. }
834. }


838. *// error*

841. absolute\_error = [];
842. relative\_error = [];
843. maxMaxValE1 = 1;
844. minMinValE1 = 0;
845. maxMaxValE2 = 1;
846. minMinValE2 = 0;
848. maxValsE1 = [];
849. minValsE1 = [];
850. maxValsE2 = [];
851. minValsE2 = [];
853. EotT = [];
855. **for** (**var** i = 0 ; i <= K ; ++i) {
856. **var** timeVector1 = [];
857. **var** timeVector2 = [];
858. **var** this\_t = t \* i;
859. **var** this\_max1 = 0;
860. **var** this\_min1 = 0;
862. **var** this\_max2 = 0;
863. **var** this\_min2 = 0;
865. **for** (**var** j = 0 ; j <= N ; ++j) {
866. **var** this\_x = xVector[j];
868. **var** err1 = solve[i][j] - etalon[i][j];
869. **var** err2 = 0;
870. **if** (Math.abs(etalon[i][j]) > 0.1) {
871. err2 = Math.abs(1 - solve[i][j]/etalon[i][j]);
872. }
873. **else** {
874. err2 = Math.abs(1 - (solve[i][j]+0.2)/(etalon[i][j] + 0.2));
875. }
877. **if** (j == 0) {
878. this\_max1 = err1;
879. this\_min1 = err1;
881. this\_max2 = err2;
882. this\_min2 = err2;
883. }
884. **else** {
885. this\_max1 = (err1 > this\_max1 ? err1 : this\_max1);
886. this\_min1 = (err1 < this\_min1 ? err1 : this\_min1);
888. this\_max2 = (err2 > this\_max2 ? err2 : this\_max2);
889. this\_min2 = (err2 < this\_min2 ? err2 : this\_min2);
890. }
891. timeVector1.push(err1);
892. timeVector2.push(err2);
893. }
895. absolute\_error.push(timeVector1);
896. relative\_error.push(timeVector2);
898. maxValsE1.push(this\_max1);
899. minValsE1.push(this\_min1);
901. maxValsE2.push(this\_max2);
902. minValsE2.push(this\_min2);

905. **if** (i == 0) {
906. maxMaxValE1 = this\_max1;
907. minMinValE1 = this\_min1;
909. maxMaxValE2 = this\_max2;
910. minMinValE2 = this\_min2;
912. **var** stepEot = Math.max(maxMaxValE1, Math.abs(minMinValE1));
914. EotT.push(stepEot);
916. minEotT = stepEot;
917. maxEotT = stepEot;
918. }
919. **else** {
920. maxMaxValE1 = (this\_max1 > maxMaxValE1 ? this\_max1 : maxMaxValE1);
921. minMinValE1 = (this\_min1 < minMinValE1 ? this\_min1 : minMinValE1);
923. **var** stepEot = Math.max(this\_max1, Math.abs(this\_min1));
925. minEotT = (stepEot < minEotT ? stepEot : minEotT);
926. maxEotT = (stepEot > maxEotT ? stepEot : maxEotT);
928. EotT.push(stepEot);
930. maxMaxValE2 = (this\_max2 > maxMaxValE2 ? this\_max2 : maxMaxValE2);
931. minMinValE2 = (this\_min2 < minMinValE2 ? this\_min2 : minMinValE2);
932. }
933. }


937. makeEtalonGraph();
938. makeSolveGraph();
939. makeErrorGraph();
940. makeErrorTGraph();
942. go = **true**;
943. }
945. **function** transfX(x) {
946. **var** newX = 60 + (567 \* (x / l));
947. **return** newX;
948. }
950. **function** transfY(y) {
951. **var** newY = 0;
952. **if** (maxVals.length > 0 && scale\_type == 2) {
953. newY = 418.6 - (382.6 \* (y - minVals[s\_cur])/(maxVals[s\_cur] - minVals[s\_cur]));
954. }
955. **else** {
956. newY = 418.6 - (382.6 \* (y - minMinVal)/(maxMaxVal - minMinVal));
957. }
958. **return** newY;
959. }

962. **function** transfEX(x) {
963. **var** newX = 60.6 + (344.2 \* (x / l));
964. **return** newX;
965. }

968. **function** transfE2X(x) {
969. **var** newX = 60.6 + (344.2 \* (x / T));
970. **return** newX;
971. }
973. **function** transfEY(y) {
974. **var** newY = 0;
975. **if** (maxValsE.length > 0 && scale\_type == 2) {
976. newY = 700 - (231 \* (y - minValsE[s\_cur])/(maxValsE[s\_cur] - minValsE[s\_cur]));
977. }
978. **else** {
979. newY = 700 - (231 \* (y - minMinValE)/(maxMaxValE - minMinValE));
980. }
981. **return** newY;
982. }
984. **function** transfE2Y(y) {
985. **var** newY = 0;
986. newY = 700 - (231 \* (y - minEotT)/(maxEotT - minEotT));
988. **return** newY;
989. }

992. **function** makeEtalonGraph() {
993. **for** (**var** i = 0 ; i < N ; ++i) {
994. **var** join = **new** lib.line();
995. stage.addChild(join);
996. join.x = transfX(xVector[i]);
997. join.y = transfY(etalon[0][i]);
998. join.endX = transfX(xVector[i+1]);
999. join.endY = transfY(etalon[0][i+1]);
1001. join.gotoAndStop(0);
1002. join.num = i;
1004. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1006. join.scaleX = join.len;
1007. join.scaleY = 1;
1009. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1011. join.gen = generation;

1014. join.visible = **true**;
1015. join.alpha = 1;
1017. join.addEventListener('tick', setPoses);
1018. }
1020. }
1022. **function** makeSolveGraph() {
1023. **for** (**var** i = 0 ; i < N ; ++i) {
1024. **var** join = **new** lib.line();
1025. stage.addChild(join);
1026. join.x = transfX(xVector[i]);
1027. join.y = transfY(solve[0][i]);
1028. join.endX = transfX(xVector[i+1]);
1029. join.endY = transfY(solve[0][i+1]);
1031. join.gotoAndStop(1);
1032. join.num = i;
1034. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1036. join.scaleX = join.len;
1037. join.scaleY = 1;
1039. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1041. join.gen = generation;

1044. join.visible = **true**;
1045. join.alpha = 1;
1047. join.addEventListener('tick', setPoses2);
1048. }
1050. }
1052. **function** makeErrorGraph() {
1053. setError();
1054. **for** (**var** i = 0 ; i < N ; ++i) {
1055. **var** join = **new** lib.line();
1056. stage.addChild(join);
1057. join.x = transfEX(xVector[i]);
1058. join.y = transfEY(errorGraph[0][i]);
1059. join.endX = transfEX(xVector[i+1]);
1060. join.endY = transfEY(errorGraph[0][i+1]);
1062. join.gotoAndStop(3);
1063. join.num = i;
1065. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1067. join.scaleX = join.len;
1068. join.scaleY = 1;
1070. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1072. join.gen = generation;

1075. join.visible = **true**;
1076. join.alpha = 1;
1078. join.addEventListener('tick', setPoses3);
1079. }
1081. }

1084. **function** makeErrorTGraph() {
1085. **for** (**var** i = 0 ; i < K ; ++i) {
1086. **var** join = **new** lib.line();
1087. stage.addChild(join);
1088. join.x = transfE2X(tVector[i]);
1089. join.y = transfE2Y(EotT[i]);
1090. join.endX = transfE2X(tVector[i+1]);
1091. join.endY = transfE2Y(tVector[i+1]);
1093. join.gotoAndStop(3);
1094. join.num = i;
1096. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1098. join.scaleX = join.len;
1099. join.scaleY = 1;
1101. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1103. join.gen = generation;

1106. join.visible = **true**;
1107. join.alpha = 1;
1109. join.addEventListener('tick', setPoses4);
1110. }
1112. }
1114. **function** setPoses(e) {
1115. **var** object = e.currentTarget;
1116. **if** (object.gen == generation) {
1117. object.x = transfX(xVector[object.num]);
1118. object.y = transfY(etalon[s\_cur][object.num]);
1119. object.endX = transfX(xVector[object.num+1]);
1120. object.endY = transfY(etalon[s\_cur][object.num+1]);
1122. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
1124. object.scaleX = object.len;
1125. object.scaleY = 1;
1127. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1128. }
1129. **else** **if** (object.gen != generation) {
1130. object.alpha -= 3/30;
1131. }
1133. **if** (object.alpha <= 0) {
1134. object.alpha = 0;
1135. object.visible = **false**;
1136. object.removeEventListener('tick', setPoses);
1137. stage.removeChild(object);
1138. }
1140. }

1143. **function** setPoses2(e) {
1144. **var** object = e.currentTarget;
1145. **if** (object.gen == generation) {
1146. object.x = transfX(xVector[object.num]);
1147. object.y = transfY(solve[s\_cur][object.num]);
1148. object.endX = transfX(xVector[object.num+1]);
1149. object.endY = transfY(solve[s\_cur][object.num+1]);
1151. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
1153. object.scaleX = object.len;
1154. object.scaleY = 1;
1156. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1157. }
1158. **else** **if** (object.gen != generation) {
1159. object.alpha -= 3/30;
1160. }
1162. **if** (object.alpha <= 0) {
1163. object.alpha = 0;
1164. object.visible = **false**;
1165. object.removeEventListener('tick', setPoses2);
1166. stage.removeChild(object);
1167. }
1169. }

1172. **function** setPoses3(e) {
1173. **var** object = e.currentTarget;
1174. **if** (object.gen == generation) {
1175. object.x = transfEX(xVector[object.num]);
1176. object.y = transfEY(errorGraph[s\_cur][object.num]);
1177. object.endX = transfEX(xVector[object.num+1]);
1178. object.endY = transfEY(errorGraph[s\_cur][object.num+1]);
1180. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
1182. object.scaleX = object.len;
1183. object.scaleY = 1;
1185. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1186. }
1187. **else** **if** (object.gen != generation) {
1188. object.alpha -= 3/30;
1189. }
1191. **if** (pogr\_type <= 2) {
1192. object.visible = **true**;
1193. }
1194. **else** {
1195. object.visible = **false**;
1196. }
1198. **if** (object.alpha <= 0) {
1199. object.alpha = 0;
1200. object.visible = **false**;
1201. object.removeEventListener('tick', setPoses3);
1202. stage.removeChild(object);
1203. }
1205. }

1208. **function** setPoses4(e) {
1209. **var** object = e.currentTarget;
1210. **if** (object.gen == generation) {
1211. object.x = transfE2X(tVector[object.num]);
1212. object.y = transfE2Y(EotT[object.num]);
1213. object.endX = transfE2X(tVector[object.num+1]);
1214. object.endY = transfE2Y(EotT[object.num+1]);
1216. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
1218. object.scaleX = object.len;
1219. object.scaleY = 1;
1221. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1222. }
1223. **else** **if** (object.gen != generation) {
1224. object.alpha -= 3/30;
1225. }
1227. **if** (pogr\_type == 3) {
1228. object.visible = **true**;
1229. }
1230. **else** {
1231. object.visible = **false**;
1232. }
1234. **if** (object.alpha <= 0) {
1235. object.alpha = 0;
1236. object.visible = **false**;
1237. object.removeEventListener('tick', setPoses4);
1238. stage.removeChild(object);
1239. }
1241. }







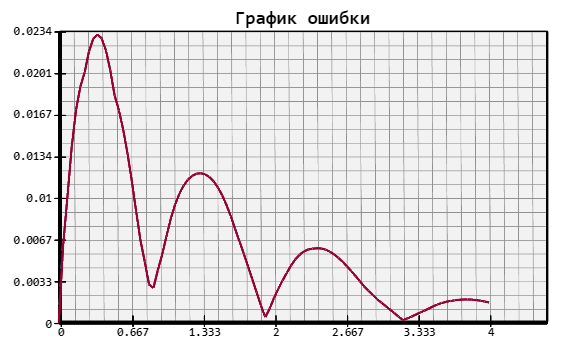
1250. *//метод прогонки*
1251. **function** progonka(matrix, vectorB) {
1252. **var** vectorX = [];
1254. **var** N = vectorB.length;
1256. **var** alphas = [];
1257. **var** betas = [];

1260. **for** (**var** i = 0 ; i < vectorB.length ; ++i) {
1261. alphas.push(0);
1262. betas.push(0);
1263. }
1265. *//Прямой ход прогонки*
1267. **for** (**var** i = 0; i < N; ++i) {
1268. **var** A0, C0, B0, F0;
1270. **if** (i - 1 < 0) {
1271. A0 = 0;
1272. }
1273. **else** {
1274. A0 = matrix[i][i - 1];
1275. }
1277. C0 = -1 \* matrix[i][i];
1279. **if** (i + 1 < N) {
1280. B0 = matrix[i][i + 1];
1281. }
1282. **else** {
1283. B0 = 0;
1284. }
1286. F0 = vectorB[i];

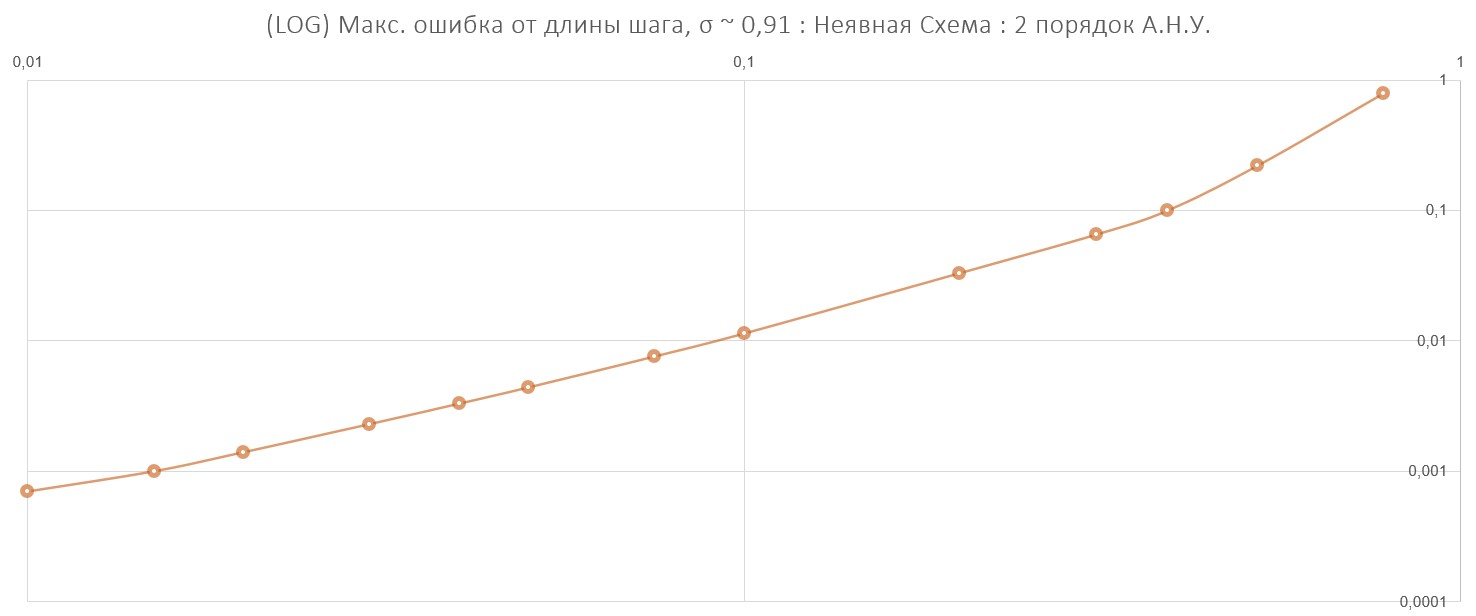
1289. **if** (i == 0) {
1290. alphas[i] = B0 / C0;
1291. betas[i] = -(F0 / C0);
1292. }
1293. **else** **if** (i == N - 1) {
1294. alphas[i] = 0;
1295. betas[i] = (betas[i - 1] \* A0 - F0) / (C0 - alphas[i - 1] \* A0);
1296. }
1297. **else** {
1298. alphas[i] = B0 / (C0 - alphas[i - 1] \* A0);
1299. betas[i] = (betas[i - 1] \* A0 - F0) / (C0 - alphas[i - 1] \* A0);
1300. }
1302. }

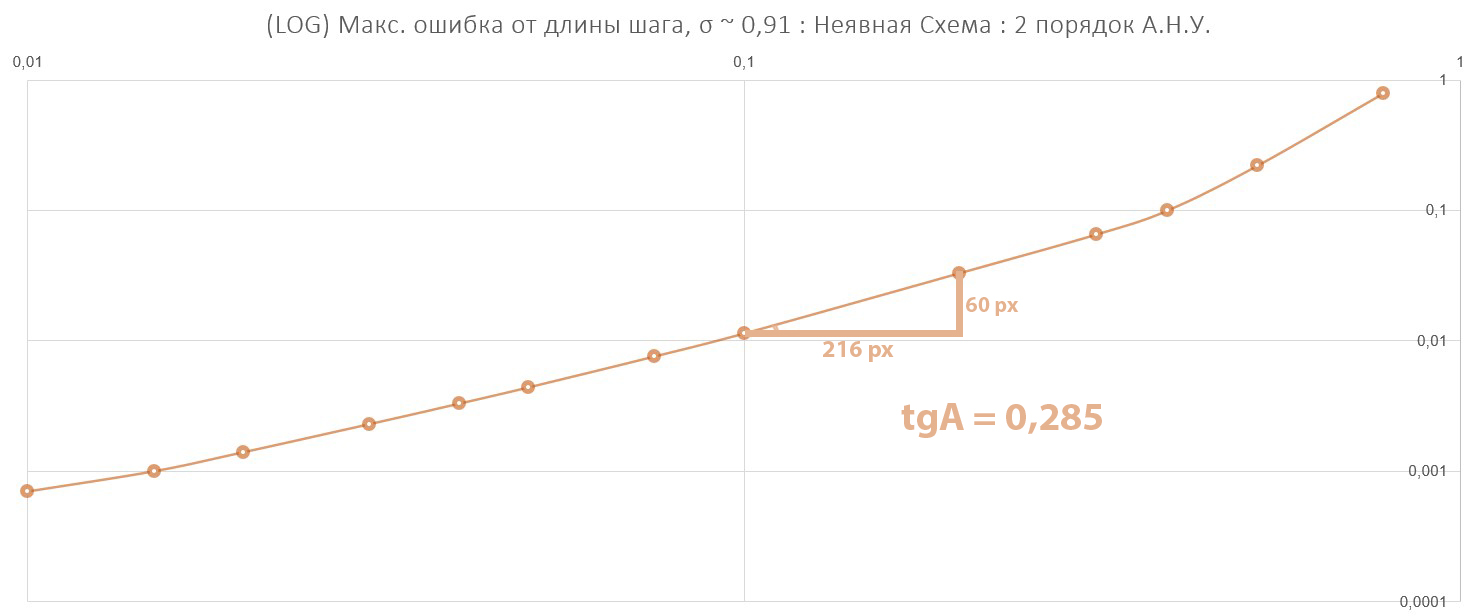
1305. vectorX[N - 1] = betas[N - 1];
1307. **for** (**var** i = 2; i <= N; ++i) {
1308. vectorX[N - i] = alphas[N - i] \* vectorX[N - i + 1] + betas[N - i];
1309. }
1311. **return** vectorX;
1312. }
1313. Графики

Для T = 4









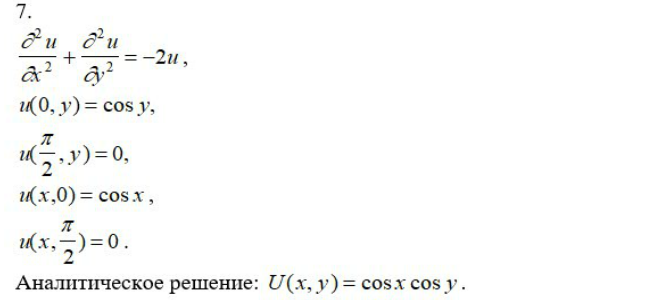
Данная лабораторная работа выполнена: **19 января 2021.**

**Лабораторная работа 7**

1. Тема ЛР:

Метод конечных разностей для решения уравнений эллиптического типа. Уравнение Пуассона. Методы Либмана и Зейделя.

1. Вариант : **7**



1. Алгоритм:

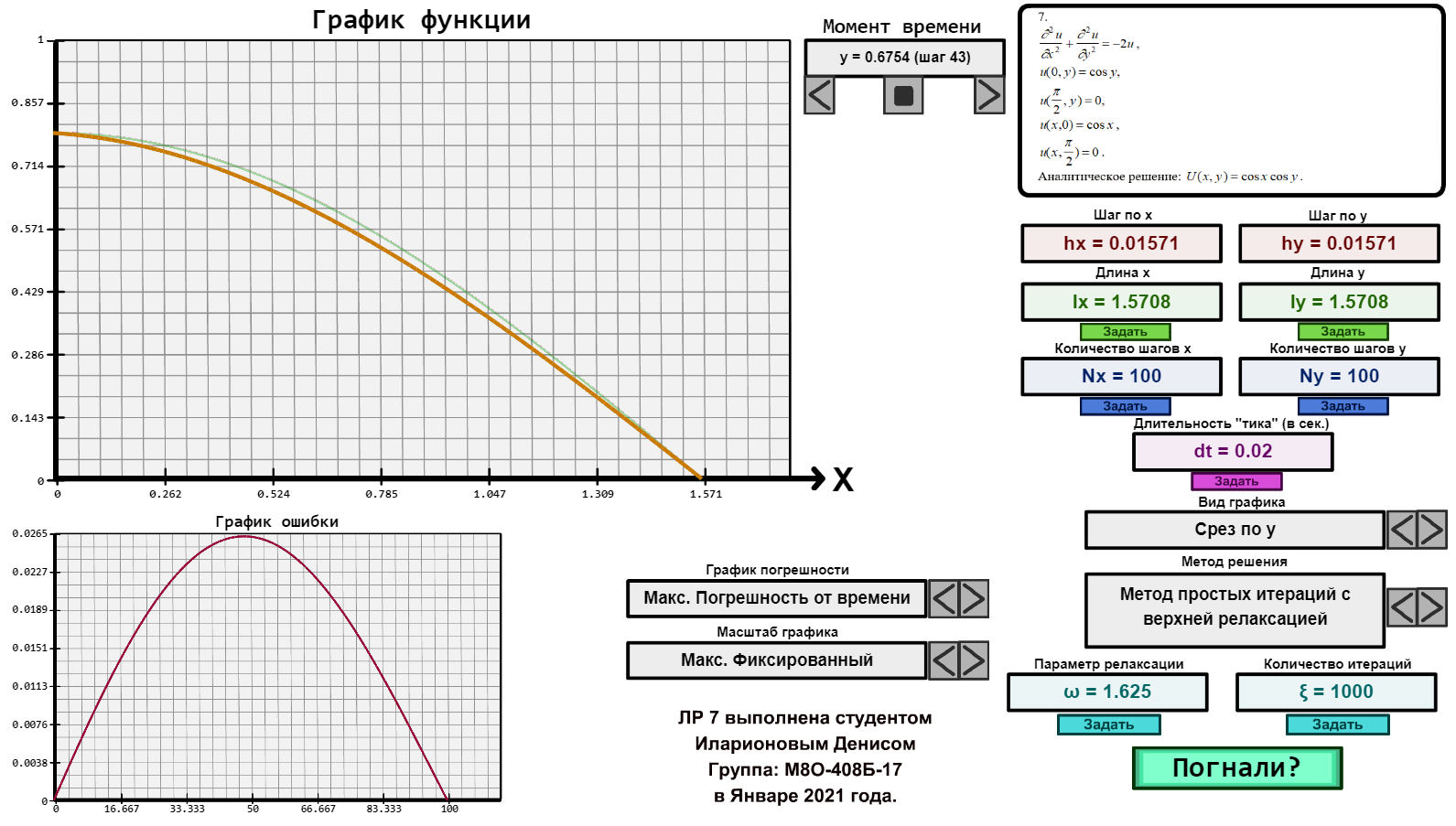
В данной лабораторной пришлось напрячься, потому что нужно отобразить срез как по Х, так и по Y. То есть нужно выбрать одну из этих координат временной. Это заняло у меня много времени и сил. А сама лабораторная не очень сложная. Тут используются методы Либмана и Зейделя, которые были еще в 1 семестре. А также параметр релаксации. При ω < 1 – метод Либмана. При ω = 1 – метод Зейделя. А при ω > 1 - метод Либмана с верхней релаксацией, который показывает наилучший результат, но параметр не должен быть близко к двойке или больше или равен ей, тогда алгоритм становится неустойчивым.

1. Среда разработки:

Adobe Animate, Javascript + HTML5

1. Реализация

Можно выбрать какую координату использовать временной + все, что в прошлых лабораторных. Границы аппроксимировать, к счастью, снова не надо (повезло с вариантом).



Вместо точности, я указал количество итераций. Думаю, так более логично и безопасно. Потому что при большом количестве шагов, чтобы добиться высокой точности необходимо огромное число итераций, что программа просто зависнет. Даже при максимальном параметре релаксации. Поэтому, количество итераций тут использовать удобнее, я считаю так.

1. Код программы
2. **var** alpha = 0; *//α*
3. **var** beta = 1; *//β*
4. **var** gamma = 0; *//γ*
5. **var** delta = 1; *//δ*
7. *//Обозначим все как граничные условия 3 рода*
9. **function** phi\_x0(y) {
10. **return** Math.cos(y);
11. }
13. **function** phi\_xl(y) {
14. **return** 0;
15. }
17. **function** phi\_y0(x) {
18. **return** Math.cos(x);
19. }
21. **function** phi\_yl(x) {
22. **return** 0;
23. }


27. **function** U(x, y) {
28. **return** Math.cos(x) \* Math.cos(y);
29. } *//Аналитическое решение*

32. *//Параметры сетки*
33. **var** lx = 3.1415926535/2; *//интервал x*
34. **var** Nx = 20; *//количество шагов x*
35. **var** Ny = 20; *//количество шагов y*
36. **var** ly = 3.1415926535/2; *//интервал y*
38. **var** hx = 0; **var** hy = 0;
40. **var** a = 1; **var** b = 2; **var** c = -3;
42. **var** s1\_cur = 0; *//по y*
43. **var** s2\_cur = 0; *//по x*
45. **var** dt = 0.02; *//длительность тика*
46. **var** cur\_tick = 0;

49. **var** generation = 0; *//номер генерации*
50. **var** go = **false**; *//увеличивать счетчик тиков?*

53. **var** setupType = 1;
54. *//1 срез по y*
55. *//2 срез по x*
57. **var** iters = 1000;
58. *//Количество итераций*

61. **this**.addEventListener("tick", det\_ht.bind(**this**));
62. **function** det\_ht() {
63. hx = lx / Nx;
64. hy = ly / Ny;
66. *//определяем шаги по времени и пространству*
68. **this**.h\_text.text = "hx = " + Math.round(hx \* 100000)/100000;
69. **this**.tau\_text.text = "hy = " + Math.round(hy \* 100000)/100000;
71. **this**.l\_text.text = "lx = " + Math.round(lx \* 100000)/100000;
72. **this**.T\_text.text = "ly = " + Math.round(ly \* 100000)/100000;
74. **this**.N\_text.text = "Nx = " + Math.round(Nx);
75. **this**.K\_text.text = "Ny = " + Math.round(Ny);
77. **this**.dt\_text.text = "dt = " + Math.round(dt\*1000)/1000;
79. **this**.om\_text.text = "ω = " + Math.round(relax\*1000)/1000;
80. **this**.it\_text.text = "ξ = " + Math.round(iters);
82. **if** (setupType == 1) {
83. **if** (go) {
84. cur\_tick += 1/60;
85. **if** (cur\_tick >= dt && s1\_cur < Ny) {
86. **var** tGot = Math.floor(cur\_tick / dt);
87. cur\_tick %= dt;
88. s1\_cur += Math.min(Ny - s1\_cur, tGot);
89. }
90. **else** **if** (s1\_cur == Ny) {
91. go = **false**;
92. }
93. }
94. }
95. **else** **if** (setupType == 2) {
96. **if** (go) {
97. cur\_tick += 1/60;
98. **if** (cur\_tick >= dt && s2\_cur < Nx) {
99. **var** tGot = Math.floor(cur\_tick / dt);
100. cur\_tick %= dt;
101. s2\_cur += Math.min(Nx - s2\_cur, tGot);
102. }
103. **else** **if** (s2\_cur == Nx) {
104. go = **false**;
105. }
106. }
107. }
109. }



114. **this**.setGreen1.addEventListener("click", set\_l\_prompt.bind(**this**));
115. **function** set\_l\_prompt() {
116. generation += 1;
117. **var** temp = prompt("Введите lx:", '');
118. temp = Number.parseFloat(temp);
119. **if** (isNaN(temp)) {
120. *//если ввели какую-то хрень*
121. temp = lx;
122. }
123. **else** **if** (temp > 1000000) {
124. *//ограничение на величину*
125. temp = 1000000;
126. }
127. **else** **if** (temp <= 0) {
128. *//всегда положительно*
129. temp = 0.001;
130. }
131. lx = temp;
132. }


136. **this**.setGreen2.addEventListener("click", set\_T\_prompt.bind(**this**));
137. **function** set\_T\_prompt() {
138. generation += 1;
139. **var** temp = prompt("Введите ly:", '');
140. temp = Number.parseFloat(temp);
141. **if** (isNaN(temp)) {
142. *//если ввели какую-то хрень*
143. temp = ly;
144. }
145. **else** **if** (temp > 1000000) {
146. *//ограничение на величину*
147. temp = 1000000;
148. }
149. **else** **if** (temp <= 0) {
150. *//всегда положительно*
151. temp = 0.001;
152. }
153. ly = temp;
154. }

157. **this**.setBlue1.addEventListener("click", set\_N\_prompt.bind(**this**));
158. **function** set\_N\_prompt() {
159. generation += 1;
160. **var** temp = prompt("Введите Nx:", '');
161. temp = Number.parseInt(temp);
162. **if** (isNaN(temp)) {
163. *//если ввели какую-то хрень*
164. temp = Nx;
165. }
166. **else** **if** (temp > 200) {
167. *//ограничение на величину*
168. temp = 200;
169. }
170. **else** **if** (temp <= 0) {
171. *//всегда положительно*
172. temp = 1;
173. }
174. Nx = temp;
175. }

178. **this**.setBlue2.addEventListener("click", set\_K\_prompt.bind(**this**));
179. **function** set\_K\_prompt() {
180. generation += 1;
181. **var** temp = prompt("Введите Ny:", '');
182. temp = Number.parseInt(temp);
183. **if** (isNaN(temp)) {
184. *//если ввели какую-то хрень*
185. temp = Ny;
186. }
187. **else** **if** (temp > 200) {
188. *//ограничение на величину*
189. temp = 200;
190. }
191. **else** **if** (temp <= 0) {
192. *//всегда положительно*
193. temp = 1;
194. }
195. Ny = temp;
196. }

199. **this**.setPurp1.addEventListener("click", set\_dt\_prompt.bind(**this**));
200. **function** set\_dt\_prompt() {
201. cur\_tick = 0;
202. **var** temp = prompt("Введите dt (для ручного переключения можно ввести очень большим):", '');
203. temp = Number.parseFloat(temp);
204. **if** (isNaN(temp)) {
205. *//если ввели какую-то хрень*
206. temp = dt;
207. }
208. **else** **if** (temp > 1000000) {
209. *//ограничение на величину*
210. temp = 1000000;
211. }
212. **else** **if** (temp < 0.02) {
213. *//всегда положительно и больше 0.2 (почти плавная смена графика)*
214. temp = 0.02;
215. }
216. dt = temp;
217. }

220. **this**.setMag1.addEventListener("click", set\_omega\_prompt.bind(**this**));
221. **function** set\_omega\_prompt() {
222. generation += 1;
223. **var** temp = prompt("Введите ω:", '');
224. temp = Number.parseFloat(temp);
225. **if** (isNaN(temp)) {
226. *//если ввели какую-то хрень*
227. temp = relax;
228. }
229. **else** **if** (temp > 1.99) {
230. *//ограничение на величину*
231. temp = 1.99;
232. }
233. **else** **if** (temp <= 0) {
234. *//всегда положительно*
235. temp = 0.001;
236. }
237. relax = temp;
238. }

241. **this**.setMag2.addEventListener("click", set\_i\_prompt.bind(**this**));
242. **function** set\_i\_prompt() {
243. generation += 1;
244. **var** temp = prompt("Введите ξ:", '');
245. temp = Number.parseInt(temp);
246. **if** (isNaN(temp)) {
247. *//если ввели какую-то хрень*
248. temp = iters;
249. }
250. **else** **if** (temp > 10000) {
251. *//ограничение на величину*
252. temp = 10000;
253. }
254. **else** **if** (temp <= 0) {
255. *//всегда положительно*
256. temp = 1;
257. }
258. iters = temp;
259. }
261. *//Тип схемы - центрально-разностная*
262. *//1,2 - Явная/Неявная схема*
264. **var** relax = 1; *//параметр релаксации*

267. **var** meth\_type = 1;
268. *//Метод решения*
269. *//1 - метод Либмана*
270. *//2 - метод Зейделя*
271. *//3 - метод Либмана с верхней релаксации*
273. **var** approx\_type = 1;
274. *//Метод аппроксимации 2 н.у*
275. *//1 - Первый порядок*
276. *//2 - Второй порядок*


280. **var** pogr\_type = 1;
281. *//Тип погрешности*
282. *//1 - Абсолютная*
283. *//2 - Относительная*
284. *//3 - От времени*
286. **var** scale\_type = 1;
287. *//Масштаб графика*
288. *//1 - Фиксированный*
289. *//2 - Динамический*
291. **this**.SLeft1.addEventListener("click", scheme\_left.bind(**this**));
292. **function** scheme\_left() {
293. **if** (setupType == 1) {
294. setupType = 2;
295. }
296. **else** {
297. setupType -= 1;
298. }
299. }
301. **this**.SRight1.addEventListener("click", scheme\_right.bind(**this**));
302. **function** scheme\_right() {
303. **if** (setupType == 2) {
304. setupType = 1;
305. }
306. **else** {
307. setupType += 1;
308. }
309. }

312. **this**.SLeft2.addEventListener("click", meth\_left.bind(**this**));
313. **function** meth\_left() {
314. **if** (meth\_type == 1) {
315. meth\_type = 3;
316. }
317. **else** {
318. meth\_type -= 1;
319. }
320. }
322. **this**.SRight2.addEventListener("click", meth\_right.bind(**this**));
323. **function** meth\_right() {
324. **if** (meth\_type == 3) {
325. meth\_type = 1;
326. }
327. **else** {
328. meth\_type += 1;
329. }
330. }


334. **this**.SLeft3.addEventListener("click", curStep\_left.bind(**this**));
335. **function** curStep\_left() {
336. **if** (setupType == 1) {
337. **if** (s1\_cur > 0) {
338. s1\_cur -= 1;
339. }
340. }
341. **else** {
342. **if** (s2\_cur > 0) {
343. s2\_cur -= 1;
344. }
345. }
346. }
348. **this**.SRight3.addEventListener("click", curStep\_right.bind(**this**));
349. **function** curStep\_right() {
350. **if** (setupType == 1) {
351. **if** (s1\_cur < Ny) {
352. s1\_cur += 1;
353. }
354. }
355. **else** {
356. **if** (s2\_cur < Nx) {
357. s2\_cur += 1;
358. }
359. }
360. }

363. **this**.stopBtn.addEventListener("click", curStep\_stop.bind(**this**));
364. **function** curStep\_stop() {
365. dt = 1000000;
366. }
368. **this**.SLeft4.addEventListener("click", pogr\_left.bind(**this**));
369. **function** pogr\_left() {
370. **if** (pogr\_type == 1) {
371. pogr\_type = 3;
372. }
373. **else** {
374. pogr\_type -= 1;
375. }
376. }
378. **this**.SRight4.addEventListener("click", pogr\_right.bind(**this**));
379. **function** pogr\_right() {
380. **if** (pogr\_type == 3) {
381. pogr\_type = 1;
382. }
383. **else** {
384. pogr\_type += 1;
385. }
386. }
388. **this**.SLeft5.addEventListener("click", scale\_left.bind(**this**));
389. **function** scale\_left() {
390. **if** (scale\_type == 1) {
391. scale\_type = 2;
392. }
393. **else** {
394. scale\_type -= 1;
395. }
396. }
398. **this**.SRight5.addEventListener("click", scale\_right.bind(**this**));
399. **function** scale\_right() {
400. **if** (scale\_type == 2) {
401. scale\_type = 1;
402. }
403. **else** {
404. scale\_type += 1;
405. }
406. }
408. **this**.addEventListener("tick", setTexts2.bind(**this**));
409. **function** setTexts2() {
410. **if** (setupType == 1) {
411. **this**.scheme\_text.text = "Срез по y";
412. theta = 1;
413. }
414. **else** **if** (setupType == 2) {
415. **this**.scheme\_text.text = "Срез по x";
416. theta = 0;
417. }
419. **if** (meth\_type == 1) {
420. **this**.meth\_text.text = "Метод Либмана";
421. }
422. **else** **if** (meth\_type == 2) {
423. **this**.meth\_text.text = "Метод Зейделя";
424. }
425. **else** **if** (meth\_type == 3) {
426. **this**.meth\_text.text = "Метод простых итераций с верхней релаксацией";
427. }

430. **if** (pogr\_type == 1) {
431. **this**.pogr\_text.text = "Абсолютная погрешность";
432. }
433. **else** **if** (pogr\_type == 2) {
434. **this**.pogr\_text.text = "Относительная погрешность";
435. }
436. **else** **if** (pogr\_type == 3) {
437. **this**.pogr\_text.text = "Макс. Погрешность от времени";
438. }

441. **if** (scale\_type == 1) {
442. **this**.scale\_text.text = "Макс. Фиксированный";
443. }
444. **else** **if** (scale\_type == 2) {
445. **this**.scale\_text.text = "Динамический";
446. }

449. **if** (setupType == 1) {
450. **this**.divDown0.text = "" + 0;
451. **this**.divDown1.text = "" + Math.round(lx\*1/6\*1000)/1000;
452. **this**.divDown2.text = "" + Math.round(lx\*2/6\*1000)/1000;
453. **this**.divDown3.text = "" + Math.round(lx\*3/6\*1000)/1000;
454. **this**.divDown4.text = "" + Math.round(lx\*4/6\*1000)/1000;
455. **this**.divDown5.text = "" + Math.round(lx\*5/6\*1000)/1000;
456. **this**.divDown6.text = "" + Math.round(lx\*1000)/1000;
457. }
458. **else** {
459. **this**.divDown0.text = "" + 0;
460. **this**.divDown1.text = "" + Math.round(ly\*1/6\*1000)/1000;
461. **this**.divDown2.text = "" + Math.round(ly\*2/6\*1000)/1000;
462. **this**.divDown3.text = "" + Math.round(ly\*3/6\*1000)/1000;
463. **this**.divDown4.text = "" + Math.round(ly\*4/6\*1000)/1000;
464. **this**.divDown5.text = "" + Math.round(ly\*5/6\*1000)/1000;
465. **this**.divDown6.text = "" + Math.round(ly\*1000)/1000;
466. }
468. **if** (setupType == 1) {
469. **if** (pogr\_type <= 2) {
470. **this**.divDown0err.text = "" + 0;
471. **this**.divDown1err.text = "" + Math.round(lx\*1/6\*1000)/1000;
472. **this**.divDown2err.text = "" + Math.round(lx\*2/6\*1000)/1000;
473. **this**.divDown3err.text = "" + Math.round(lx\*3/6\*1000)/1000;
474. **this**.divDown4err.text = "" + Math.round(lx\*4/6\*1000)/1000;
475. **this**.divDown5err.text = "" + Math.round(lx\*5/6\*1000)/1000;
476. **this**.divDown6err.text = "" + Math.round(lx\*1000)/1000;
477. }
478. **else** {
479. **this**.divDown0err.text = "" + 0;
480. **this**.divDown1err.text = "" + Math.round(Nx\*1/6\*1000)/1000;
481. **this**.divDown2err.text = "" + Math.round(Nx\*2/6\*1000)/1000;
482. **this**.divDown3err.text = "" + Math.round(Nx\*3/6\*1000)/1000;
483. **this**.divDown4err.text = "" + Math.round(Nx\*4/6\*1000)/1000;
484. **this**.divDown5err.text = "" + Math.round(Nx\*5/6\*1000)/1000;
485. **this**.divDown6err.text = "" + Math.round(Nx\*1000)/1000;
486. }
487. }
488. **else** {
489. **if** (pogr\_type <= 2) {
490. **this**.divDown0err.text = "" + 0;
491. **this**.divDown1err.text = "" + Math.round(ly\*1/6\*1000)/1000;
492. **this**.divDown2err.text = "" + Math.round(ly\*2/6\*1000)/1000;
493. **this**.divDown3err.text = "" + Math.round(ly\*3/6\*1000)/1000;
494. **this**.divDown4err.text = "" + Math.round(ly\*4/6\*1000)/1000;
495. **this**.divDown5err.text = "" + Math.round(ly\*5/6\*1000)/1000;
496. **this**.divDown6err.text = "" + Math.round(ly\*1000)/1000;
497. }
498. **else** {
499. **this**.divDown0err.text = "" + 0;
500. **this**.divDown1err.text = "" + Math.round(Ny\*1/6\*1000)/1000;
501. **this**.divDown2err.text = "" + Math.round(Ny\*2/6\*1000)/1000;
502. **this**.divDown3err.text = "" + Math.round(Ny\*3/6\*1000)/1000;
503. **this**.divDown4err.text = "" + Math.round(Ny\*4/6\*1000)/1000;
504. **this**.divDown5err.text = "" + Math.round(Ny\*5/6\*1000)/1000;
505. **this**.divDown6err.text = "" + Math.round(Ny\*1000)/1000;
506. }
507. }
509. **if** (setupType == 1) {
510. **if** (maxValsx.length > 0 && scale\_type == 2) {
511. **this**.divUp0.text = "" + Math.round(minValsx[s1\_cur]\*1000)/1000;
512. **this**.divUp1.text = "" + Math.round((minValsx[s1\_cur] + (maxValsx[s1\_cur] - minValsx[s1\_cur])\*1/7)\*1000)/1000;
513. **this**.divUp2.text = "" + Math.round((minValsx[s1\_cur] + (maxValsx[s1\_cur] - minValsx[s1\_cur])\*2/7)\*1000)/1000;
514. **this**.divUp3.text = "" + Math.round((minValsx[s1\_cur] + (maxValsx[s1\_cur] - minValsx[s1\_cur])\*3/7)\*1000)/1000;
515. **this**.divUp4.text = "" + Math.round((minValsx[s1\_cur] + (maxValsx[s1\_cur] - minValsx[s1\_cur])\*4/7)\*1000)/1000;
516. **this**.divUp5.text = "" + Math.round((minValsx[s1\_cur] + (maxValsx[s1\_cur] - minValsx[s1\_cur])\*5/7)\*1000)/1000;
517. **this**.divUp6.text = "" + Math.round((minValsx[s1\_cur] + (maxValsx[s1\_cur] - minValsx[s1\_cur])\*6/7)\*1000)/1000;
518. **this**.divUp7.text = "" + Math.round(maxValsx[s1\_cur]\*1000)/1000;
519. }
520. **else** {
521. **this**.divUp0.text = "" + Math.round(minMinValx\*1000)/1000;
522. **this**.divUp1.text = "" + Math.round((minMinValx + (maxMaxValx - minMinValx)\*1/7)\*1000)/1000;
523. **this**.divUp2.text = "" + Math.round((minMinValx + (maxMaxValx - minMinValx)\*2/7)\*1000)/1000;
524. **this**.divUp3.text = "" + Math.round((minMinValx + (maxMaxValx - minMinValx)\*3/7)\*1000)/1000;
525. **this**.divUp4.text = "" + Math.round((minMinValx + (maxMaxValx - minMinValx)\*4/7)\*1000)/1000;
526. **this**.divUp5.text = "" + Math.round((minMinValx + (maxMaxValx - minMinValx)\*5/7)\*1000)/1000;
527. **this**.divUp6.text = "" + Math.round((minMinValx + (maxMaxValx - minMinValx)\*6/7)\*1000)/1000;
528. **this**.divUp7.text = "" + Math.round(maxMaxValx\*1000)/1000;
529. }
530. }
531. **else** {
532. **if** (maxValsy.length > 0 && scale\_type == 2) {
533. **this**.divUp0.text = "" + Math.round(minValsy[s2\_cur]\*1000)/1000;
534. **this**.divUp1.text = "" + Math.round((minValsy[s2\_cur] + (maxValsy[s2\_cur] - minValsy[s2\_cur])\*1/7)\*1000)/1000;
535. **this**.divUp2.text = "" + Math.round((minValsy[s2\_cur] + (maxValsy[s2\_cur] - minValsy[s2\_cur])\*2/7)\*1000)/1000;
536. **this**.divUp3.text = "" + Math.round((minValsy[s2\_cur] + (maxValsy[s2\_cur] - minValsy[s2\_cur])\*3/7)\*1000)/1000;
537. **this**.divUp4.text = "" + Math.round((minValsy[s2\_cur] + (maxValsy[s2\_cur] - minValsy[s2\_cur])\*4/7)\*1000)/1000;
538. **this**.divUp5.text = "" + Math.round((minValsy[s2\_cur] + (maxValsy[s2\_cur] - minValsy[s2\_cur])\*5/7)\*1000)/1000;
539. **this**.divUp6.text = "" + Math.round((minValsy[s2\_cur] + (maxValsy[s2\_cur] - minValsy[s2\_cur])\*6/7)\*1000)/1000;
540. **this**.divUp7.text = "" + Math.round(maxValsy[s2\_cur]\*1000)/1000;
541. }
542. **else** {
543. **this**.divUp0.text = "" + Math.round(minMinValy\*1000)/1000;
544. **this**.divUp1.text = "" + Math.round((minMinValy + (maxMaxValy - minMinValy)\*1/7)\*1000)/1000;
545. **this**.divUp2.text = "" + Math.round((minMinValy + (maxMaxValy - minMinValy)\*2/7)\*1000)/1000;
546. **this**.divUp3.text = "" + Math.round((minMinValy + (maxMaxValy - minMinValy)\*3/7)\*1000)/1000;
547. **this**.divUp4.text = "" + Math.round((minMinValy + (maxMaxValy - minMinValy)\*4/7)\*1000)/1000;
548. **this**.divUp5.text = "" + Math.round((minMinValy + (maxMaxValy - minMinValy)\*5/7)\*1000)/1000;
549. **this**.divUp6.text = "" + Math.round((minMinValy + (maxMaxValy - minMinValy)\*6/7)\*1000)/1000;
550. **this**.divUp7.text = "" + Math.round(maxMaxValy\*1000)/1000;
551. }
552. }
554. **if** (setupType == 1) {
555. **if** (maxValsE.length > 0 && scale\_type == 2 && pogr\_type <= 2) {
556. **this**.divUp0err.text = "" + Math.round(minValsE[s1\_cur]\*1000)/1000;
557. **this**.divUp1err.text = "" + Math.round((minValsE[s1\_cur] + (maxValsE[s1\_cur] - minValsE[s1\_cur])\*1/7)\*1000)/1000;
558. **this**.divUp2err.text = "" + Math.round((minValsE[s1\_cur] + (maxValsE[s1\_cur] - minValsE[s1\_cur])\*2/7)\*1000)/1000;
559. **this**.divUp3err.text = "" + Math.round((minValsE[s1\_cur] + (maxValsE[s1\_cur] - minValsE[s1\_cur])\*3/7)\*1000)/1000;
560. **this**.divUp4err.text = "" + Math.round((minValsE[s1\_cur] + (maxValsE[s1\_cur] - minValsE[s1\_cur])\*4/7)\*1000)/1000;
561. **this**.divUp5err.text = "" + Math.round((minValsE[s1\_cur] + (maxValsE[s1\_cur] - minValsE[s1\_cur])\*5/7)\*1000)/1000;
562. **this**.divUp6err.text = "" + Math.round((minValsE[s1\_cur] + (maxValsE[s1\_cur] - minValsE[s1\_cur])\*6/7)\*1000)/1000;
563. **this**.divUp7err.text = "" + Math.round(maxValsE[s1\_cur]\*1000)/1000;
564. }
565. **else** **if** (pogr\_type <= 2) {
566. **this**.divUp0err.text = "" + Math.round(minMinValE\*1000)/1000;
567. **this**.divUp1err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*1/7)\*1000)/1000;
568. **this**.divUp2err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*2/7)\*1000)/1000;
569. **this**.divUp3err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*3/7)\*1000)/1000;
570. **this**.divUp4err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*4/7)\*1000)/1000;
571. **this**.divUp5err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*5/7)\*1000)/1000;
572. **this**.divUp6err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*6/7)\*1000)/1000;
573. **this**.divUp7err.text = "" + Math.round(maxMaxValE\*1000)/1000;
574. }
575. **else** {
576. **this**.divUp0err.text = "" + Math.round(minEotTx\*10000)/10000;
577. **this**.divUp1err.text = "" + Math.round((minEotTx + (maxEotTx - minEotTx)\*1/7)\*10000)/10000;
578. **this**.divUp2err.text = "" + Math.round((minEotTx + (maxEotTx - minEotTx)\*2/7)\*10000)/10000;
579. **this**.divUp3err.text = "" + Math.round((minEotTx + (maxEotTx - minEotTx)\*3/7)\*10000)/10000;
580. **this**.divUp4err.text = "" + Math.round((minEotTx + (maxEotTx - minEotTx)\*4/7)\*10000)/10000;
581. **this**.divUp5err.text = "" + Math.round((minEotTx + (maxEotTx - minEotTx)\*5/7)\*10000)/10000;
582. **this**.divUp6err.text = "" + Math.round((minEotTx + (maxEotTx - minEotTx)\*6/7)\*10000)/10000;
583. **this**.divUp7err.text = "" + Math.round(maxEotTx\*10000)/10000;
584. }
585. }
586. **else** {
587. **if** (maxValsE.length > 0 && scale\_type == 2 && pogr\_type <= 2) {
588. **this**.divUp0err.text = "" + Math.round(minValsE[s2\_cur]\*1000)/1000;
589. **this**.divUp1err.text = "" + Math.round((minValsE[s2\_cur] + (maxValsE[s2\_cur] - minValsE[s2\_cur])\*1/7)\*1000)/1000;
590. **this**.divUp2err.text = "" + Math.round((minValsE[s2\_cur] + (maxValsE[s2\_cur] - minValsE[s2\_cur])\*2/7)\*1000)/1000;
591. **this**.divUp3err.text = "" + Math.round((minValsE[s2\_cur] + (maxValsE[s2\_cur] - minValsE[s2\_cur])\*3/7)\*1000)/1000;
592. **this**.divUp4err.text = "" + Math.round((minValsE[s2\_cur] + (maxValsE[s2\_cur] - minValsE[s2\_cur])\*4/7)\*1000)/1000;
593. **this**.divUp5err.text = "" + Math.round((minValsE[s2\_cur] + (maxValsE[s2\_cur] - minValsE[s2\_cur])\*5/7)\*1000)/1000;
594. **this**.divUp6err.text = "" + Math.round((minValsE[s2\_cur] + (maxValsE[s2\_cur] - minValsE[s2\_cur])\*6/7)\*1000)/1000;
595. **this**.divUp7err.text = "" + Math.round(maxValsE[s2\_cur]\*1000)/1000;
596. }
597. **else** **if** (pogr\_type <= 2) {
598. **this**.divUp0err.text = "" + Math.round(minMinValE\*1000)/1000;
599. **this**.divUp1err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*1/7)\*1000)/1000;
600. **this**.divUp2err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*2/7)\*1000)/1000;
601. **this**.divUp3err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*3/7)\*1000)/1000;
602. **this**.divUp4err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*4/7)\*1000)/1000;
603. **this**.divUp5err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*5/7)\*1000)/1000;
604. **this**.divUp6err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*6/7)\*1000)/1000;
605. **this**.divUp7err.text = "" + Math.round(maxMaxValE\*1000)/1000;
606. }
607. **else** {
608. **this**.divUp0err.text = "" + Math.round(minEotTy\*10000)/10000;
609. **this**.divUp1err.text = "" + Math.round((minEotTy + (maxEotTy - minEotTy)\*1/7)\*10000)/10000;
610. **this**.divUp2err.text = "" + Math.round((minEotTy + (maxEotTy - minEotTy)\*2/7)\*10000)/10000;
611. **this**.divUp3err.text = "" + Math.round((minEotTy + (maxEotTy - minEotTy)\*3/7)\*10000)/10000;
612. **this**.divUp4err.text = "" + Math.round((minEotTy + (maxEotTy - minEotTy)\*4/7)\*10000)/10000;
613. **this**.divUp5err.text = "" + Math.round((minEotTy + (maxEotTy - minEotTy)\*5/7)\*10000)/10000;
614. **this**.divUp6err.text = "" + Math.round((minEotTy + (maxEotTy - minEotTy)\*6/7)\*10000)/10000;
615. **this**.divUp7err.text = "" + Math.round(maxEotTy\*10000)/10000;
616. }
617. }

620. **if** (setupType == 1) {
621. s1\_cur2 = s1\_cur\*hy;
622. **this**.step\_text.text = "y = " + Math.round(s1\_cur2\*10000)/10000 + " (шаг " + Math.round(s1\_cur) + ")";
623. **this**.axisDown.axisTp.text = "X";
624. }
625. **else** {
626. s2\_cur2 = s2\_cur\*hx;
627. **this**.step\_text.text = "x = " + Math.round(s2\_cur2\*10000)/10000 + " (шаг " + Math.round(s2\_cur) + ")";
628. **this**.axisDown.axisTp.text = "Y";
629. }
630. **this**.axisDown.y = Math.min(Math.max(transfY(0), 26.4), 418.6);

633. **this**.axisDownErr.y = Math.min(Math.max(transfEY(0), 468), 700);
634. }

637. *//baseGraph*
638. *//x = 0 : posX = 60*
639. *//y = 0 : posY = 418.6*
641. *//x = max : posX = 627*
642. *//y = max : posY = 26.4*
644. *//errGraph*
645. *//x = 0 : posX = 53.6*
646. *//y = 0 : posY = 700*
648. *//x = max : posX = 397.8*
649. *//y = max : posY = 468.4*




655. **var** s1\_cur = 0;
656. *//текущий шаг по y*
658. **var** s2\_cur = 0;
659. *//текущий шаг по x*

662. **var** etalon = [];
663. **var** xVector = [];
664. **var** yVector = [];
666. **var** maxMaxValx = 1;
667. **var** minMinValx = 0;
669. **var** maxMaxValy = 1;
670. **var** minMinValy = 0;
672. **var** maxMaxValE = 1; *//link*
673. **var** minMinValE = 0; *//link*
675. **var** maxMaxValE1x = 1;
676. **var** minMinValE1x = 0;
677. **var** maxMaxValE2x = 1;
678. **var** minMinValE2x = 0;
680. **var** maxMaxValE1y = 1;
681. **var** minMinValE1y = 0;
682. **var** maxMaxValE2y = 1;
683. **var** minMinValE2y = 0;
685. **var** maxValsx = [];
686. **var** minValsx = [];
688. **var** maxValsy = [];
689. **var** minValsy = [];
691. **var** maxValsE = []; *//link*
692. **var** minValsE = []; *//link*
694. **var** maxValsE1x = [];
695. **var** minValsE1x = [];
696. **var** maxValsE2x = [];
697. **var** minValsE2x = [];
699. **var** maxValsE1y = [];
700. **var** minValsE1y = [];
701. **var** maxValsE2y = [];
702. **var** minValsE2y = [];


706. **var** EotT = [];
708. **this**.addEventListener("tick", setError.bind(**this**));
709. **function** setError() {
710. **if** (setupType == 1) {
711. **if** (pogr\_type == 1) {
712. maxMaxValE = maxMaxValE1x;
713. minMinValE = minMinValE1x;
714. maxValsE = maxValsE1x;
715. minValsE = minValsE1x;
716. errorGraphX = absolute\_errorX;
717. errorGraphY = absolute\_errorY;
718. }
719. **else** {
720. maxMaxValE = maxMaxValE2x;
721. minMinValE = minMinValE2x;
722. maxValsE = maxValsE2x;
723. minValsE = minValsE2x;
724. errorGraphX = relative\_errorX;
725. errorGraphY = relative\_errorY;
726. }
727. }
728. **else** {
729. **if** (pogr\_type == 1) {
730. maxMaxValE = maxMaxValE1y;
731. minMinValE = minMinValE1y;
732. maxValsE = maxValsE1y;
733. minValsE = minValsE1y;
734. errorGraphX = absolute\_errorX;
735. errorGraphY = absolute\_errorY;
736. }
737. **else** {
738. maxMaxValE = maxMaxValE2y;
739. minMinValE = minMinValE2y;
740. maxValsE = maxValsE2y;
741. minValsE = minValsE2y;
742. errorGraphX = relative\_errorX;
743. errorGraphY = relative\_errorY;
744. }
745. }
746. }
748. **var** solve = [];
749. **var** solve2 = [];

752. **function** copyV(vector) {
753. **var** newV = [];
754. **for** (**var** i = 0 ; i < vector.length ; ++i) {
755. newV.push(vector[i]);
756. }
757. **return** newV;
758. }
760. **function** copyM(matrix) {
761. **var** newM = [];
762. **for** (**var** i = 0 ; i < matrix.length ; ++i) {
763. newM.push(copyV(matrix[i]));
764. }
765. **return** newM;
766. }
768. **var** absolute\_errorX = [];
769. **var** relative\_errorX = [];
771. **var** absolute\_errorY = [];
772. **var** relative\_errorY = [];
773. **var** errorGraphX = [];
774. **var** errorGraphY = [];
776. **var** minEotTx = 0;
777. **var** maxEotTx = 1;
779. **var** minEotTy = 0;
780. **var** maxEotTy = 1;




786. **this**.beginBtn.addEventListener("click", beginSimulation.bind(**this**));
787. **function** beginSimulation() {
788. generation += 1;
789. s1\_cur = 0;
790. s2\_cur = 0;
791. etalon = [];
792. solve = [];
793. solve2 = [];
794. maxMaxValx = 1;
795. minMinValx = 0;
797. maxMaxValy = 1;
798. minMinValy = 0;
800. maxValsx = [];
801. minValsx = [];
803. maxValsy = [];
804. minValsy = [];
806. xVector = [];
807. yVector = [];
809. **for** (**var** j = 0 ; j <= Nx ; ++j) {
810. xVector.push(hx \* j);
811. }
812. **for** (**var** j = 0 ; j <= Ny ; ++j) {
813. yVector.push(hy \* j);
814. }
815. **for** (**var** i = 0 ; i <= Ny ; ++i) {
816. **var** timeVector = [];
817. **var** this\_y = yVector[i];
818. yVector.push(this\_y);
819. **var** this\_maxX = 0;
820. **var** this\_minX = 0;
821. **for** (**var** j = 0 ; j <= Nx ; ++j) {
822. **var** this\_x = xVector[j];
823. **var** func\_res = U(this\_x, this\_y);
824. **if** (j == 0) {
825. this\_maxX = func\_res;
826. this\_minX = func\_res;
827. }
828. **else** {
829. this\_maxX = (func\_res > this\_maxX ? func\_res : this\_maxX);
830. this\_minX = (func\_res < this\_minX ? func\_res : this\_minX);
831. }
832. **if** (i == 0) {
833. maxValsy.push(func\_res);
834. minValsy.push(func\_res);
835. }
836. **else** {
837. maxValsy[j] = (func\_res > maxValsy[j] ? func\_res : maxValsy[j]);
838. minValsy[j] = (func\_res < minValsy[j] ? func\_res : minValsy[j]);
839. }

842. timeVector.push(func\_res);
843. }
844. etalon.push(timeVector);
845. maxValsx.push(this\_maxX);
846. minValsx.push(this\_minX);
847. **if** (i == 0) {
848. maxMaxValx = this\_maxX;
849. minMinValx = this\_minX;
850. }
851. **else** {
852. maxMaxValx = (this\_maxX > maxMaxValx ? this\_maxX : maxMaxValx);
853. minMinValx = (this\_minX < minMinValx ? this\_minX : minMinValx);
854. }
855. }
857. **for** (**var** i = 0 ; i < maxValsy.length ; ++i) {
858. **if** (i == 0) {
859. maxMaxValy = maxValsy[i];
860. minMinValy = minValsy[i];
861. }
862. **else** {
863. maxMaxValy = (maxValsy[i] > maxMaxValy ? maxValsy[i] : maxMaxValy);
864. minMinValy = (minValsy[i] < minMinValy ? minValsy[i] : minMinValy);
865. }
866. }
868. **for** (**var** p = 0 ; p <= Ny ; ++p) {
869. **var** solvation = [];
870. **var** curY = p\*hy;
871. **for** (**var** q = 0 ; q <= Nx ; ++q) {
872. **var** curX = q\*hx;
873. **if** (p == 0) {
874. solvation.push(phi\_y0(curX));
875. }
876. **else** **if** (p == Ny) {
877. solvation.push(phi\_yl(curX));
878. }
879. **else** {
880. **if** (q == 0) {
881. solvation.push(phi\_x0(curY));
882. }
883. **else** **if** (q == Nx) {
884. solvation.push(phi\_xl(curY));
885. }
886. **else** {
887. solvation.push(0);
888. }
889. }
890. }
891. solve.push(copyV(solvation));
892. solve2.push(copyV(solvation));
893. }


897. **if** (meth\_type == 1) {
898. **for** (**var** it = 0 ; it < iters ; ++it) {
899. **for** (**var** i = 1 ; i < Ny ; ++i) {
900. **for** (**var** j = 1 ; j < Nx ; ++j) {
901. solve2[i][j] = ((solve[i+1][j] + solve[i-1][j])\*hy\*hy
902. + (solve[i][j+1] + solve[i][j-1])\*hx\*hx) / (2\*hx\*hx + 2\*hy\*hy - 2\*hx\*hx\*hy\*hy);
903. }
904. }
905. solve = copyM(solve2);
906. }
907. }

910. **if** (meth\_type == 2) {
911. **for** (**var** it = 0 ; it < iters ; ++it) {
912. **for** (**var** i = 1 ; i < Ny ; ++i) {
913. **for** (**var** j = 1 ; j < Nx ; ++j) {
914. solve[i][j] = ((solve[i+1][j] + solve[i-1][j])\*hy\*hy
915. + (solve[i][j+1] + solve[i][j-1])\*hx\*hx) / (2\*hx\*hx + 2\*hy\*hy - 2\*hx\*hx\*hy\*hy);
916. }
917. }
918. }
919. }
921. **if** (meth\_type == 3) {
922. **for** (**var** it = 0 ; it < iters ; ++it) {
923. **for** (**var** i = 1 ; i < Ny ; ++i) {
924. **for** (**var** j = 1 ; j < Nx ; ++j) {
925. solve2[i][j] = ((solve[i+1][j] + solve[i-1][j])\*hy\*hy
926. + (solve[i][j+1] + solve[i][j-1])\*hx\*hx) / (2\*hx\*hx + 2\*hy\*hy - 2\*hx\*hx\*hy\*hy);
928. solve[i][j] = relax\*solve2[i][j] + (1-relax)\*solve[i][j];
929. }
930. }
931. }
932. }
934. **for** (**var** i = 0 ; i < maxValsx.length ; ++i) {
935. **for** (**var** j = 0 ; j < maxValsy.length ; ++j) {
936. maxValsx[i] = (solve[i][j] > maxValsx[i] ? solve[i][j] : maxValsx[i]);
937. minValsx[i] = (solve[i][j] < minValsx[i] ? solve[i][j] : minValsx[i]);
938. }
939. maxMaxValx = (maxValsx[i] > maxMaxValx ? maxValsx[i] : maxMaxValx);
940. minMinValx = (minValsx[i] < minMinValx ? minValsx[i] : minMinValx);
941. }
943. **for** (**var** i = 0 ; i < maxValsy.length ; ++i) {
944. **for** (**var** j = 0 ; j < maxValsx.length ; ++j) {
945. maxValsy[i] = (solve[j][i] > maxValsy[i] ? solve[j][i] : maxValsy[i]);
946. minValsy[i] = (solve[j][i] < minValsy[i] ? solve[j][i] : minValsy[i]);
947. }
948. maxMaxValy = (maxValsy[i] > maxMaxValy ? maxValsy[i] : maxMaxValx);
949. minMinValy = (minValsy[i] < minMinValy ? minValsy[i] : minMinValx);
950. }


954. *// error*

957. absolute\_errorX = [];
958. relative\_errorX = [];
960. absolute\_errorY = [];
961. relative\_errorY = [];
963. maxMaxValE1x = 1;
964. minMinValE1x = 0;
965. maxMaxValE2x = 1;
966. minMinValE2x = 0;
968. maxMaxValE1y = 1;
969. minMinValE1y = 0;
970. maxMaxValE2y = 1;
971. minMinValE2y = 0;
973. maxValsE1x = [];
974. minValsE1x = [];
975. maxValsE2x = [];
976. minValsE2x = [];
978. maxValsE1y = [];
979. minValsE1y = [];
980. maxValsE2y = [];
981. minValsE2y = [];
983. EotTx = [];
984. EotTy = [];

987. *//error when y slice*
988. **for** (**var** i = 0 ; i <= Ny ; ++i) {
989. **var** timeVector1 = [];
990. **var** timeVector2 = [];
991. **var** this\_y = yVector[i]
992. **var** this\_max1 = 0;
993. **var** this\_min1 = 0;
995. **var** this\_max2 = 0;
996. **var** this\_min2 = 0;
998. **for** (**var** j = 0 ; j <= Nx ; ++j) {
999. **var** this\_x = xVector[j];
1001. **var** err1 = solve[i][j] - etalon[i][j];
1002. **var** err2 = 0;
1003. **if** (Math.abs(etalon[i][j]) > 0.1) {
1004. err2 = Math.abs(1 - solve[i][j]/etalon[i][j]);
1005. }
1006. **else** {
1007. err2 = Math.abs(1 - (solve[i][j]+0.2)/(etalon[i][j] + 0.2));
1008. }
1010. **if** (j == 0) {
1011. this\_max1 = err1;
1012. this\_min1 = err1;
1014. this\_max2 = err2;
1015. this\_min2 = err2;
1016. }
1017. **else** {
1018. this\_max1 = (err1 > this\_max1 ? err1 : this\_max1);
1019. this\_min1 = (err1 < this\_min1 ? err1 : this\_min1);
1021. this\_max2 = (err2 > this\_max2 ? err2 : this\_max2);
1022. this\_min2 = (err2 < this\_min2 ? err2 : this\_min2);
1023. }
1024. timeVector1.push(err1);
1025. timeVector2.push(err2);
1026. }
1028. absolute\_errorX.push(timeVector1);
1029. relative\_errorX.push(timeVector2);
1031. maxValsE1x.push(this\_max1);
1032. minValsE1x.push(this\_min1);
1034. maxValsE2x.push(this\_max2);
1035. minValsE2x.push(this\_min2);

1038. **if** (i == 0) {
1039. maxMaxValE1x = this\_max1;
1040. minMinValE1x = this\_min1;
1042. maxMaxValE2x = this\_max2;
1043. minMinValE2x = this\_min2;
1045. **var** stepEotX = Math.max(maxMaxValE1x, Math.abs(minMinValE1x));
1047. EotTx.push(stepEotX);
1049. minEotTx = stepEotX;
1050. maxEotTx = stepEotX;
1051. }
1052. **else** {
1053. maxMaxValE1x = (this\_max1 > maxMaxValE1x ? this\_max1 : maxMaxValE1x);
1054. minMinValE1x = (this\_min1 < minMinValE1x ? this\_min1 : minMinValE1x);
1056. **var** stepEotX = Math.max(this\_max1, Math.abs(this\_min1));
1058. minEotTx = (stepEotX < minEotTx ? stepEotX : minEotTx);
1059. maxEotTx = (stepEotX > maxEotTx ? stepEotX : maxEotTx);
1061. EotTx.push(stepEotX);
1063. maxMaxValE2x = (this\_max2 > maxMaxValE2x ? this\_max2 : maxMaxValE2x);
1064. minMinValE2x = (this\_min2 < minMinValE2x ? this\_min2 : minMinValE2x);
1065. }
1066. }
1068. *//error when x slice*
1069. **for** (**var** i = 0 ; i <= Nx ; ++i) {
1070. **var** timeVector1 = [];
1071. **var** timeVector2 = [];
1072. **var** this\_x = xVector[i]
1073. **var** this\_max1 = 0;
1074. **var** this\_min1 = 0;
1076. **var** this\_max2 = 0;
1077. **var** this\_min2 = 0;
1079. **for** (**var** j = 0 ; j <= Ny ; ++j) {
1080. **var** this\_y = yVector[j];
1082. **var** err1 = solve[j][i] - etalon[j][i];
1083. **var** err2 = 0;
1084. **if** (Math.abs(etalon[j][i]) > 0.1) {
1085. err2 = Math.abs(1 - solve[j][i]/etalon[j][i]);
1086. }
1087. **else** {
1088. err2 = Math.abs(1 - (solve[j][i]+0.2)/(etalon[j][i] + 0.2));
1089. }
1091. **if** (j == 0) {
1092. this\_max1 = err1;
1093. this\_min1 = err1;
1095. this\_max2 = err2;
1096. this\_min2 = err2;
1097. }
1098. **else** {
1099. this\_max1 = (err1 > this\_max1 ? err1 : this\_max1);
1100. this\_min1 = (err1 < this\_min1 ? err1 : this\_min1);
1102. this\_max2 = (err2 > this\_max2 ? err2 : this\_max2);
1103. this\_min2 = (err2 < this\_min2 ? err2 : this\_min2);
1104. }
1105. timeVector1.push(err1);
1106. timeVector2.push(err2);
1107. }
1109. absolute\_errorY.push(timeVector1);
1110. relative\_errorY.push(timeVector2);
1112. maxValsE1y.push(this\_max1);
1113. minValsE1y.push(this\_min1);
1115. maxValsE2y.push(this\_max2);
1116. minValsE2y.push(this\_min2);

1119. **if** (i == 0) {
1120. maxMaxValE1y = this\_max1;
1121. minMinValE1y = this\_min1;
1123. maxMaxValE2y = this\_max2;
1124. minMinValE2y = this\_min2;
1126. **var** stepEotY = Math.max(maxMaxValE1y, Math.abs(minMinValE1y));
1128. EotTy.push(stepEotY);
1130. minEotTy = stepEotY;
1131. maxEotTy = stepEotY;
1132. }
1133. **else** {
1134. maxMaxValE1y = (this\_max1 > maxMaxValE1y ? this\_max1 : maxMaxValE1y);
1135. minMinValE1y = (this\_min1 < minMinValE1y ? this\_min1 : minMinValE1y);
1137. **var** stepEotY = Math.max(this\_max1, Math.abs(this\_min1));
1139. minEotTy = (stepEotY < minEotTy ? stepEotY : minEotTy);
1140. maxEotTy = (stepEotY > maxEotTy ? stepEotY : maxEotTy);
1142. EotTy.push(stepEotY);
1144. maxMaxValE2y = (this\_max2 > maxMaxValE2y ? this\_max2 : maxMaxValE2y);
1145. minMinValE2y = (this\_min2 < minMinValE2y ? this\_min2 : minMinValE2y);
1146. }
1147. }
1149. makeEtalonGraph();
1150. makeSolveGraph();
1151. makeErrorGraph();
1152. makeErrorTGraph();
1154. go = **true**;
1155. }
1157. **function** transfX(x) {
1158. **var** newX;
1159. **if** (setupType == 1) {
1160. newX = 60 + (567 \* (x / lx));
1161. }
1162. **else** {
1163. newX = 60 + (567 \* (x / ly));
1164. }
1165. **return** newX;
1166. }
1168. **function** transfY(y) {
1169. **var** newY = 0;
1170. **if** (setupType == 1) {
1171. **if** (maxValsx.length > 0 && scale\_type == 2) {
1172. newY = 418.6 - (382.6 \* (y - minValsx[s1\_cur])/(maxValsx[s1\_cur] - minValsx[s1\_cur]));
1173. }
1174. **else** {
1175. newY = 418.6 - (382.6 \* (y - minMinValx)/(maxMaxValx - minMinValx));
1176. }
1177. }
1178. **else** {
1179. **if** (maxValsy.length > 0 && scale\_type == 2) {
1180. newY = 418.6 - (382.6 \* (y - minValsy[s2\_cur])/(maxValsy[s2\_cur] - minValsy[s2\_cur]));
1181. }
1182. **else** {
1183. newY = 418.6 - (382.6 \* (y - minMinValy)/(maxMaxValy - minMinValy));
1184. }
1185. }
1186. **return** newY;
1187. }

1190. **function** transfEX(x) {
1191. **var** newX;
1192. **if** (setupType == 1) {
1193. newX = 60.6 + (344.2 \* (x / lx));
1194. }
1195. **else** {
1196. newX = 60.6 + (344.2 \* (x / ly));
1197. }
1198. **return** newX;
1199. }

1202. **function** transfE2X(x) {
1203. **var** newX;
1204. **if** (setupType == 1) {
1205. newX = 60.6 + (344.2 \* (x / ly));
1206. }
1207. **else** {
1208. newX = 60.6 + (344.2 \* (x / lx));
1209. }
1210. **return** newX;
1211. }
1213. **function** transfEY(y) {
1214. **var** newY = 0;
1215. **if** (setupType == 1) {
1216. **if** (maxValsE.length > 0 && scale\_type == 2) {
1217. newY = 700 - (231 \* (y - minValsE[s1\_cur])/(maxValsE[s1\_cur] - minValsE[s1\_cur]));
1218. }
1219. **else** {
1220. newY = 700 - (231 \* (y - minMinValE)/(maxMaxValE - minMinValE));
1221. }
1222. }
1223. **else** {
1224. **if** (maxValsE.length > 0 && scale\_type == 2) {
1225. newY = 700 - (231 \* (y - minValsE[s2\_cur])/(maxValsE[s2\_cur] - minValsE[s2\_cur]));
1226. }
1227. **else** {
1228. newY = 700 - (231 \* (y - minMinValE)/(maxMaxValE - minMinValE));
1229. }
1230. }
1231. **return** newY;
1232. }
1234. **function** transfE2Y(y) {
1235. **var** newY = 0;
1236. **if** (setupType == 1) {
1237. newY = 700 - (231 \* (y - minEotTx)/(maxEotTx - minEotTx));
1238. }
1239. **else** {
1240. newY = 700 - (231 \* (y - minEotTy)/(maxEotTy - minEotTy));
1241. }
1243. **return** newY;
1244. }

1247. **function** makeEtalonGraph() {
1248. **for** (**var** i = 0 ; i < Nx ; ++i) {
1249. **var** join = **new** lib.line();
1250. stage.addChild(join);
1251. join.x = transfX(xVector[i]);
1252. join.y = transfY(etalon[0][i]);
1253. join.endX = transfX(xVector[i+1]);
1254. join.endY = transfY(etalon[0][i+1]);
1256. join.gotoAndStop(0);
1257. join.num = i;
1259. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1261. join.scaleX = join.len;
1262. join.scaleY = 1;
1264. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1266. join.gen = generation;

1269. join.visible = **true**;
1270. join.alpha = 1;
1272. join.addEventListener('tick', setPoses11);
1273. }
1274. **for** (**var** i = 0 ; i < Ny ; ++i) {
1275. **var** join = **new** lib.line();
1276. stage.addChild(join);
1277. join.x = transfX(yVector[i]);
1278. join.y = transfY(etalon[i][0]);
1279. join.endX = transfX(yVector[i+1]);
1280. join.endY = transfY(etalon[i+1][0]);
1282. join.gotoAndStop(0);
1283. join.num = i;
1285. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1287. join.scaleX = join.len;
1288. join.scaleY = 1;
1290. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1292. join.gen = generation;

1295. join.visible = **true**;
1296. join.alpha = 1;
1298. join.addEventListener('tick', setPoses12);
1299. }
1301. }
1303. **function** makeSolveGraph() {
1304. **for** (**var** i = 0 ; i < Nx ; ++i) {
1305. **var** join = **new** lib.line();
1306. stage.addChild(join);
1307. join.x = transfX(xVector[i]);
1308. join.y = transfY(solve[0][i]);
1309. join.endX = transfX(xVector[i+1]);
1310. join.endY = transfY(solve[0][i+1]);
1312. join.gotoAndStop(1);
1313. join.num = i;
1315. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1317. join.scaleX = join.len;
1318. join.scaleY = 1;
1320. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1322. join.gen = generation;

1325. join.visible = **true**;
1326. join.alpha = 1;
1328. join.addEventListener('tick', setPoses21);
1329. }
1330. **for** (**var** i = 0 ; i < Ny ; ++i) {
1331. **var** join = **new** lib.line();
1332. stage.addChild(join);
1333. join.x = transfX(yVector[i]);
1334. join.y = transfY(solve[i][0]);
1335. join.endX = transfX(yVector[i+1]);
1336. join.endY = transfY(solve[i+1][0]);
1338. join.gotoAndStop(1);
1339. join.num = i;
1341. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1343. join.scaleX = join.len;
1344. join.scaleY = 1;
1346. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1348. join.gen = generation;

1351. join.visible = **true**;
1352. join.alpha = 1;
1354. join.addEventListener('tick', setPoses22);
1355. }
1356. }
1358. **function** makeErrorGraph() {
1359. setError();
1360. **for** (**var** i = 0 ; i < Nx ; ++i) {
1361. **var** join = **new** lib.line();
1362. stage.addChild(join);
1363. join.x = transfEX(xVector[i]);
1364. join.y = transfEY(errorGraphX[0][i]);
1365. join.endX = transfEX(xVector[i+1]);
1366. join.endY = transfEY(errorGraphX[0][i+1]);
1368. join.gotoAndStop(3);
1369. join.num = i;
1371. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1373. join.scaleX = join.len;
1374. join.scaleY = 1;
1376. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1378. join.gen = generation;

1381. join.visible = **true**;
1382. join.alpha = 1;
1384. join.addEventListener('tick', setPoses31);
1385. }
1387. **for** (**var** i = 0 ; i < Ny ; ++i) {
1388. **var** join = **new** lib.line();
1389. stage.addChild(join);
1390. join.x = transfEX(yVector[i]);
1391. join.y = transfEY(errorGraphY[0][i]);
1392. join.endX = transfEX(yVector[i+1]);
1393. join.endY = transfEY(errorGraphY[0][i+1]);
1395. join.gotoAndStop(3);
1396. join.num = i;
1398. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1400. join.scaleX = join.len;
1401. join.scaleY = 1;
1403. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1405. join.gen = generation;

1408. join.visible = **true**;
1409. join.alpha = 1;
1411. join.addEventListener('tick', setPoses32);
1412. }

1415. }

1418. **function** makeErrorTGraph() {
1419. **for** (**var** i = 0 ; i < Ny ; ++i) {
1420. **var** join = **new** lib.line();
1421. stage.addChild(join);
1422. join.x = transfE2X(yVector[i]);
1423. join.y = transfE2Y(EotTx[i]);
1424. join.endX = transfE2X(yVector[i+1]);
1425. join.endY = transfE2Y(EotTx[i+1]);
1427. join.gotoAndStop(3);
1428. join.num = i;
1430. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1432. join.scaleX = join.len;
1433. join.scaleY = 1;
1435. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1437. join.gen = generation;

1440. join.visible = **true**;
1441. join.alpha = 1;
1443. join.addEventListener('tick', setPoses41);
1444. }
1445. **for** (**var** i = 0 ; i < Nx ; ++i) {
1446. **var** join = **new** lib.line();
1447. stage.addChild(join);
1448. join.x = transfE2X(xVector[i]);
1449. join.y = transfE2Y(EotTy[i]);
1450. join.endX = transfE2X(xVector[i+1]);
1451. join.endY = transfE2Y(EotTy[i+1]);
1453. join.gotoAndStop(3);
1454. join.num = i;
1456. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
1458. join.scaleX = join.len;
1459. join.scaleY = 1;
1461. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
1463. join.gen = generation;

1466. join.visible = **true**;
1467. join.alpha = 1;
1469. join.addEventListener('tick', setPoses42);
1470. }
1471. }


1475. **function** setPoses11(e) {
1476. **var** object = e.currentTarget;
1477. **if** (object.gen == generation) {
1478. object.x = transfX(xVector[object.num]);
1479. object.y = transfY(etalon[s1\_cur][object.num]);
1480. object.endX = transfX(xVector[object.num+1]);
1481. object.endY = transfY(etalon[s1\_cur][object.num+1]);
1483. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
1485. **if** (setupType == 1) {
1486. object.visible = **true**;
1487. }
1488. **else** {
1489. object.visible = **false**;
1490. }

1493. object.scaleX = object.len;
1494. object.scaleY = 1;
1496. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1497. }
1498. **else** **if** (object.gen != generation) {
1499. object.alpha -= 3/30;
1500. }
1502. **if** (object.alpha <= 0) {
1503. object.alpha = 0;
1504. object.visible = **false**;
1505. object.removeEventListener('tick', setPoses11);
1506. stage.removeChild(object);
1507. }
1509. }
1511. **function** setPoses12(e) {
1512. **var** object = e.currentTarget;
1513. **if** (object.gen == generation) {
1514. object.x = transfX(yVector[object.num]);
1515. object.y = transfY(etalon[object.num][s2\_cur]);
1516. object.endX = transfX(yVector[object.num+1]);
1517. object.endY = transfY(etalon[object.num+1][s2\_cur]);
1519. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
1521. **if** (setupType == 2) {
1522. object.visible = **true**;
1523. }
1524. **else** {
1525. object.visible = **false**;
1526. }

1529. object.scaleX = object.len;
1530. object.scaleY = 1;
1532. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1533. }
1534. **else** **if** (object.gen != generation) {
1535. object.alpha -= 3/30;
1536. }
1538. **if** (object.alpha <= 0) {
1539. object.alpha = 0;
1540. object.visible = **false**;
1541. object.removeEventListener('tick', setPoses12);
1542. stage.removeChild(object);
1543. }
1545. }
1547. **function** setPoses21(e) {
1548. **var** object = e.currentTarget;
1549. **if** (object.gen == generation) {
1550. object.x = transfX(xVector[object.num]);
1551. object.y = transfY(solve[s1\_cur][object.num]);
1552. object.endX = transfX(xVector[object.num+1]);
1553. object.endY = transfY(solve[s1\_cur][object.num+1]);
1555. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
1557. **if** (setupType == 1) {
1558. object.visible = **true**;
1559. }
1560. **else** {
1561. object.visible = **false**;
1562. }
1564. object.scaleX = object.len;
1565. object.scaleY = 1;
1567. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1568. }
1569. **else** **if** (object.gen != generation) {
1570. object.alpha -= 3/30;
1571. }
1573. **if** (object.alpha <= 0) {
1574. object.alpha = 0;
1575. object.visible = **false**;
1576. object.removeEventListener('tick', setPoses21);
1577. stage.removeChild(object);
1578. }
1580. }
1582. **function** setPoses22(e) {
1583. **var** object = e.currentTarget;
1584. **if** (object.gen == generation) {
1585. object.x = transfX(yVector[object.num]);
1586. object.y = transfY(solve[object.num][s2\_cur]);
1587. object.endX = transfX(yVector[object.num+1]);
1588. object.endY = transfY(solve[object.num+1][s2\_cur]);
1590. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
1592. **if** (setupType == 2) {
1593. object.visible = **true**;
1594. }
1595. **else** {
1596. object.visible = **false**;
1597. }

1600. object.scaleX = object.len;
1601. object.scaleY = 1;
1603. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1604. }
1605. **else** **if** (object.gen != generation) {
1606. object.alpha -= 3/30;
1607. }
1609. **if** (object.alpha <= 0) {
1610. object.alpha = 0;
1611. object.visible = **false**;
1612. object.removeEventListener('tick', setPoses22);
1613. stage.removeChild(object);
1614. }
1616. }

1619. **function** setPoses31(e) {
1620. **var** object = e.currentTarget;
1621. **if** (object.gen == generation) {
1622. object.x = transfEX(xVector[object.num]);
1623. object.y = transfEY(errorGraphX[s1\_cur][object.num]);
1624. object.endX = transfEX(xVector[object.num+1]);
1625. object.endY = transfEY(errorGraphX[s1\_cur][object.num+1]);
1627. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
1629. object.scaleX = object.len;
1630. object.scaleY = 1;
1632. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1633. }
1634. **else** **if** (object.gen != generation) {
1635. object.alpha -= 3/30;
1636. }
1638. **if** (pogr\_type <= 2) {
1639. **if** (setupType == 1) {
1640. object.visible = **true**;
1641. }
1642. **else** {
1643. object.visible = **false**;
1644. }
1645. }
1646. **else** {
1647. object.visible = **false**;
1648. }
1650. **if** (object.alpha <= 0) {
1651. object.alpha = 0;
1652. object.visible = **false**;
1653. object.removeEventListener('tick', setPoses31);
1654. stage.removeChild(object);
1655. }
1657. }

1660. **function** setPoses32(e) {
1661. **var** object = e.currentTarget;
1662. **if** (object.gen == generation) {
1663. object.x = transfEX(yVector[object.num]);
1664. object.y = transfEY(errorGraphY[s2\_cur][object.num]);
1665. *//object.y = transfEY(errorGraph[object.num][s2\_cur]);*
1666. object.endX = transfEX(yVector[object.num+1]);
1667. object.endY = transfEY(errorGraphY[s2\_cur][object.num+1]);
1668. *//object.endY = transfEY(errorGraph[object.num+1][s2\_cur]);*
1670. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));


1674. object.scaleX = object.len;
1675. object.scaleY = 1;
1677. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1678. }
1679. **else** **if** (object.gen != generation) {
1680. object.alpha -= 3/30;
1681. }
1683. **if** (pogr\_type <= 2) {
1684. **if** (setupType == 2) {
1685. object.visible = **true**;
1686. }
1687. **else** {
1688. object.visible = **false**;
1689. }
1690. }
1691. **else** {
1692. object.visible = **false**;
1693. }
1695. **if** (object.alpha <= 0) {
1696. object.alpha = 0;
1697. object.visible = **false**;
1698. object.removeEventListener('tick', setPoses32);
1699. stage.removeChild(object);
1700. }
1702. }

1705. **function** setPoses41(e) {
1706. **var** object = e.currentTarget;
1707. **if** (object.gen == generation) {
1708. object.x = transfE2X(yVector[object.num]);
1709. object.y = transfE2Y(EotTx[object.num]);
1710. object.endX = transfE2X(yVector[object.num+1]);
1711. object.endY = transfE2Y(EotTx[object.num+1]);
1713. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
1715. **if** (setupType == 1) {
1716. object.visible = **true**;
1717. }
1718. **else** {
1719. object.visible = **false**;
1720. }
1722. object.scaleX = object.len;
1723. object.scaleY = 1;
1725. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1726. }
1727. **else** **if** (object.gen != generation) {
1728. object.alpha -= 3/30;
1729. }
1731. **if** (pogr\_type == 3) {
1732. **if** (setupType == 1) {
1733. object.visible = **true**;
1734. }
1735. **else** {
1736. object.visible = **false**;
1737. }
1738. }
1739. **else** {
1740. object.visible = **false**;
1741. }
1743. **if** (object.alpha <= 0) {
1744. object.alpha = 0;
1745. object.visible = **false**;
1746. object.removeEventListener('tick', setPoses41);
1747. stage.removeChild(object);
1748. }
1750. }


1754. **function** setPoses42(e) {
1755. **var** object = e.currentTarget;
1756. **if** (object.gen == generation) {
1757. object.x = transfE2X(xVector[object.num]);
1758. object.y = transfE2Y(EotTy[object.num]);
1759. object.endX = transfE2X(xVector[object.num+1]);
1760. object.endY = transfE2Y(EotTy[object.num+1]);
1762. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));


1766. object.scaleX = object.len;
1767. object.scaleY = 1;
1769. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
1770. }
1771. **else** **if** (object.gen != generation) {
1772. object.alpha -= 3/30;
1773. }
1775. **if** (pogr\_type == 3) {
1776. **if** (setupType == 2) {
1777. object.visible = **true**;
1778. }
1779. **else** {
1780. object.visible = **false**;
1781. }
1782. }
1783. **else** {
1784. object.visible = **false**;
1785. }
1787. **if** (object.alpha <= 0) {
1788. object.alpha = 0;
1789. object.visible = **false**;
1790. object.removeEventListener('tick', setPoses42);
1791. stage.removeChild(object);
1792. }
1794. }





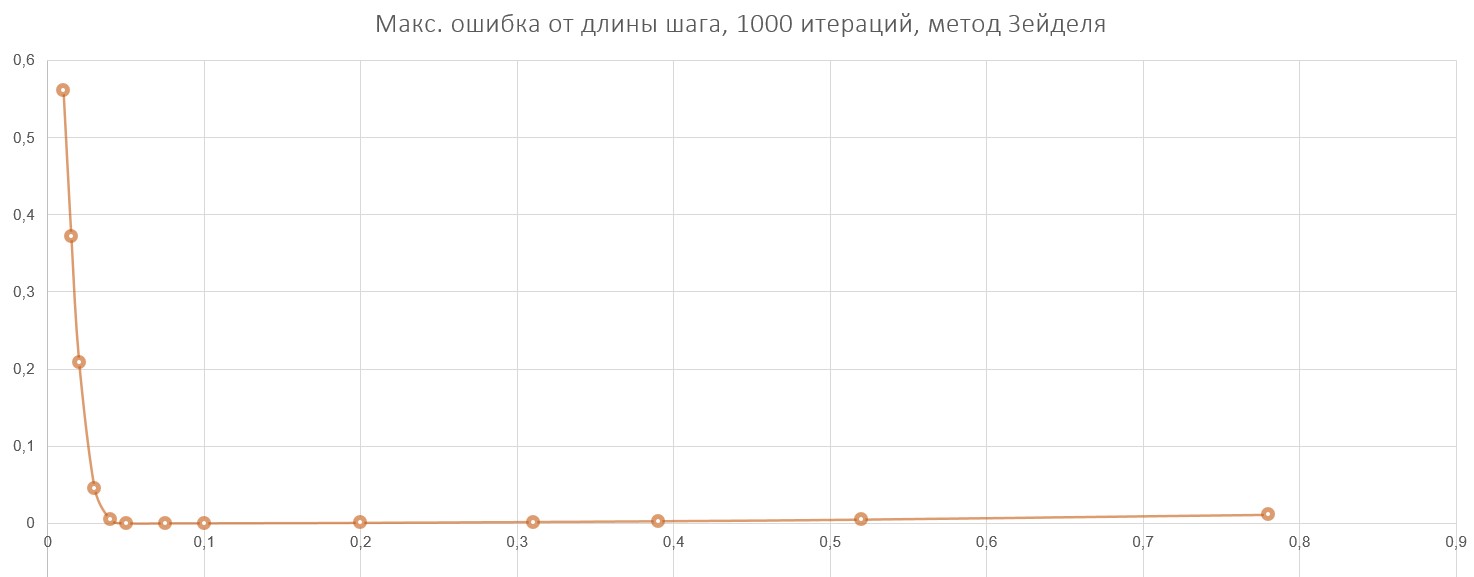

1802. *//метод прогонки*
1803. **function** progonka(matrix, vectorB) {
1804. **var** vectorX = [];
1806. **var** N = vectorB.length;
1808. **var** alphas = [];
1809. **var** betas = [];

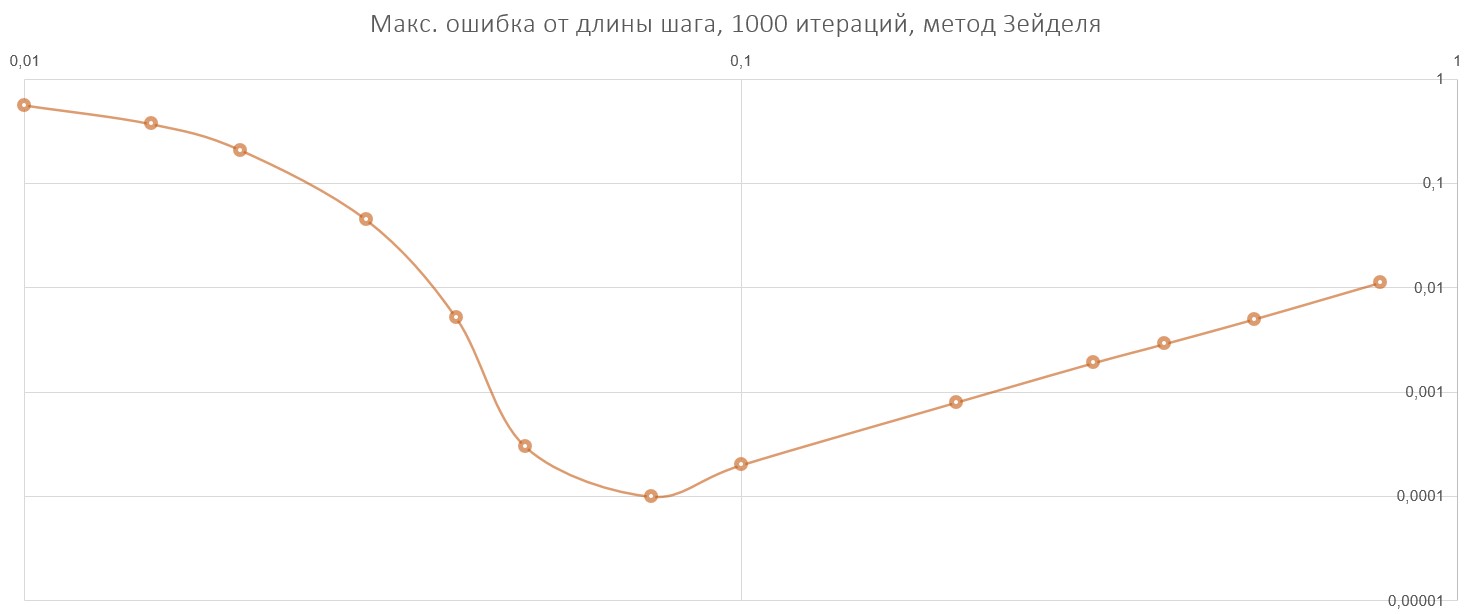
1812. **for** (**var** i = 0 ; i < vectorB.length ; ++i) {
1813. alphas.push(0);
1814. betas.push(0);
1815. }
1817. *//Прямой ход прогонки*
1819. **for** (**var** i = 0; i < N; ++i) {
1820. **var** A0, C0, B0, F0;
1822. **if** (i - 1 < 0) {
1823. A0 = 0;
1824. }
1825. **else** {
1826. A0 = matrix[i][i - 1];
1827. }
1829. C0 = -1 \* matrix[i][i];
1831. **if** (i + 1 < N) {
1832. B0 = matrix[i][i + 1];
1833. }
1834. **else** {
1835. B0 = 0;
1836. }
1838. F0 = vectorB[i];

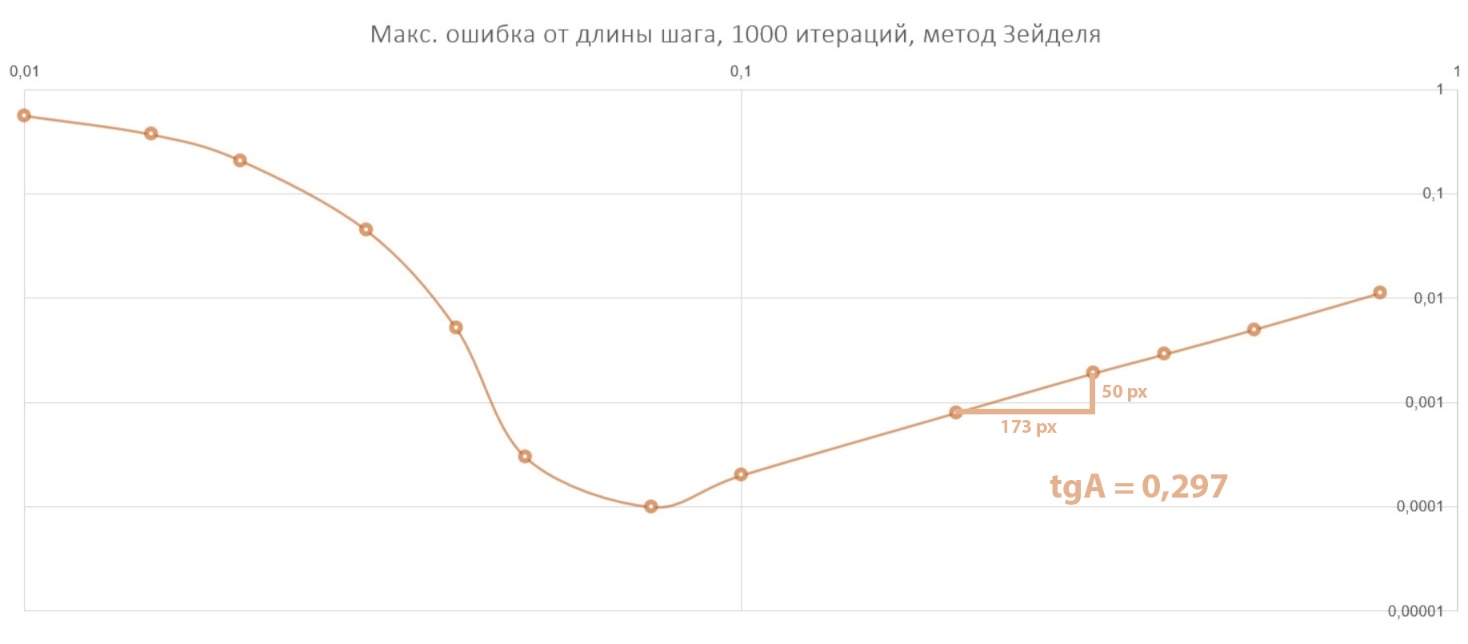
1841. **if** (i == 0) {
1842. alphas[i] = B0 / C0;
1843. betas[i] = -(F0 / C0);
1844. }
1845. **else** **if** (i == N - 1) {
1846. alphas[i] = 0;
1847. betas[i] = (betas[i - 1] \* A0 - F0) / (C0 - alphas[i - 1] \* A0);
1848. }
1849. **else** {
1850. alphas[i] = B0 / (C0 - alphas[i - 1] \* A0);
1851. betas[i] = (betas[i - 1] \* A0 - F0) / (C0 - alphas[i - 1] \* A0);
1852. }
1854. }

1857. vectorX[N - 1] = betas[N - 1];
1859. **for** (**var** i = 2; i <= N; ++i) {
1860. vectorX[N - i] = alphas[N - i] \* vectorX[N - i + 1] + betas[N - i];
1861. }
1863. **return** vectorX;
1864. }
1865. Графики









Ошибка в начале при маленьком шаге очень велика, потому что количество шагов очень большое, а 1000 итераций не успевают обработать данную задачу. Потом, когда количество шагов становится оптимальным, график достигает минимума. Затем снова растет, но уже из-за того, что шаги становятся больше. Также при разных количествах шагов по измерениям, ошибка довольно высока. Думаю, это связано с особенностью метода решения.

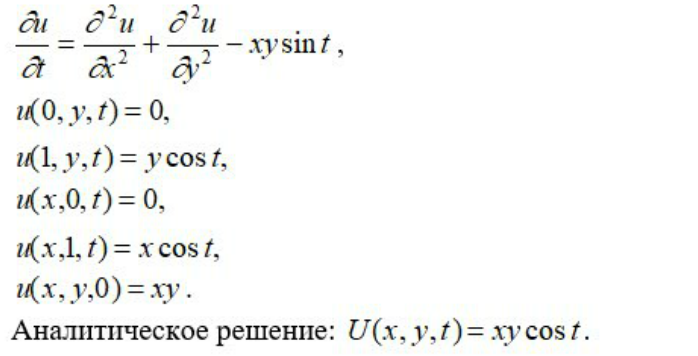
Данная лабораторная работа выполнена: **20 января 2021.**

**Лабораторная работа 8**

1. Тема ЛР:

Метод конечных разностей решения многомерных задач математической физики. Методы расщепления. Схемы дробных шагов и переменных направлений.

1. Вариант : **7**



1. Алгоритм:

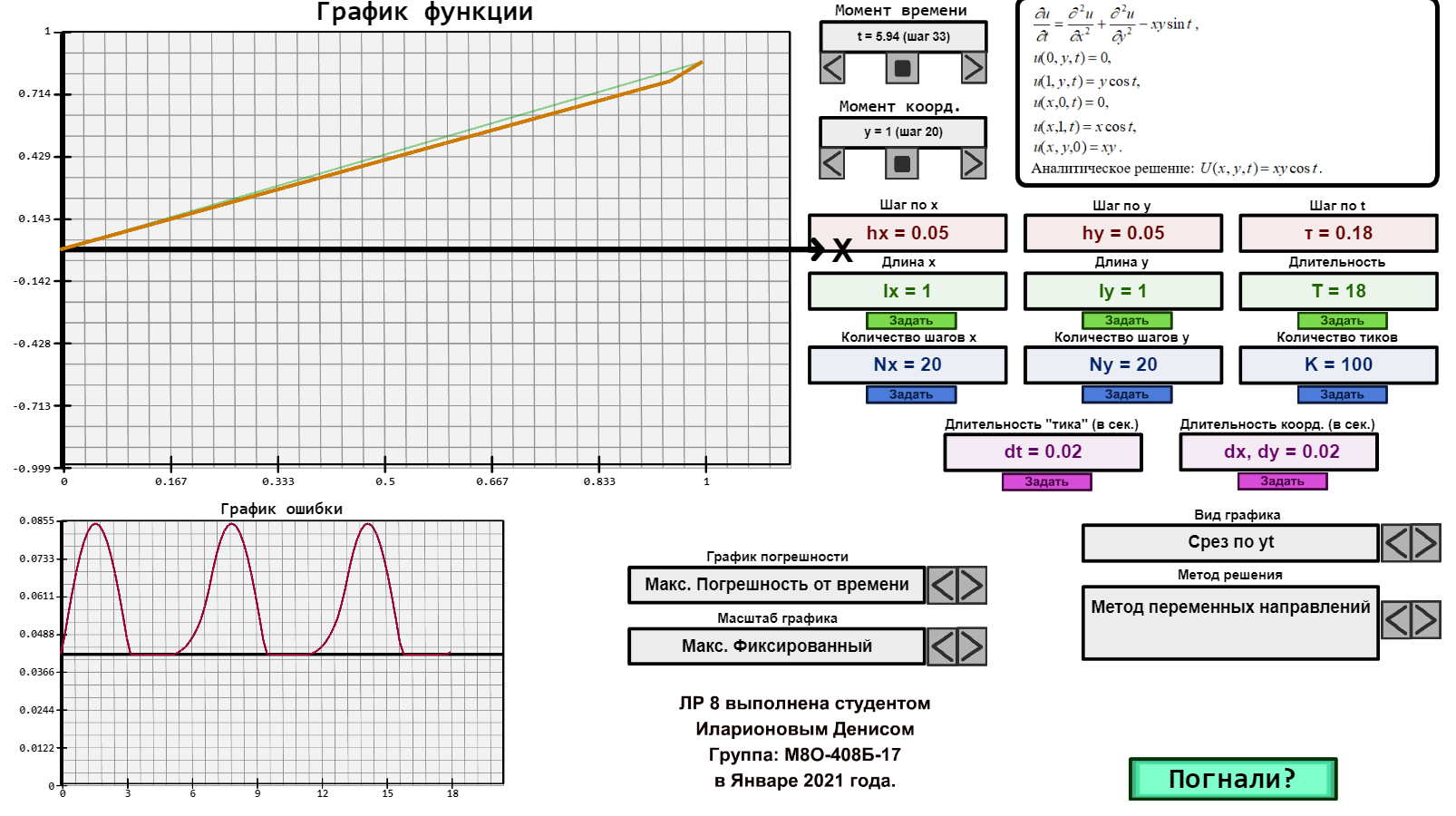
Как по мне, так это самая сложная лабораторная из всех, примерно, как курсовая. На нее ушло более 12 часов непрерывной работы и было написано почти 3 тысячи строк кода. Сама лабораторная не такая сложная на самом деле, как работа, которую пришлось мне проделать. Трехмерная сетка, выбор временных координат, куча графиков, все это мне пришлось реализовать. Схемы тоже довольно сложные, из-за того, что я не рассматривал границы, они работали у меня некорректно. Но я вовремя это исправил, хотя и сидел над этим несколько часов. Обе схемы используют похожий принцип. Мы добавляем промежуточный временной шаг t + h/2. И в зависимости от четности шага выбираем по какой переменной у нас будет явная схема, а по какой неявная (в методе дробных шагов только неявная, более того, он работает даже лучше и дает меньше погрешность). С погрешностью тоже довольно трудно было, я решил переработать ее и сделать куб погрешностей и уже с него смотреть грани. Погрешность идет максимальная по обеим временным координатам при фиксированной точке.

1. Среда разработки:

Adobe Animate, Javascript + HTML5

1. Реализация

Выбор срезов, вид графика погрешностей. Убрал относительную погрешность, не вижу в ней смысла, да и работала она криво при значениях близких к нулю. На конце погрешность может быть равна нулю. Это связано с тем, что есть некоторые граничные условия. К счастью, они снова были заданы явно и мне не пришлось аппроксимировать их как в первой лабораторной. Но и без этого ушло немало усилий и трудов.



1. Код программы
2. **var** alpha = 0; *//α*
3. **var** beta = 1; *//β*
4. **var** gamma = 0; *//γ*
5. **var** delta = 1; *//δ*
7. *//Обозначим все как граничные условия 3 рода*
9. **function** phi\_x0(y, t) {
10. **return** 0;
11. }
13. **function** phi\_xl(y, t) {
14. **return** y\*Math.cos(t);
15. }
17. **function** phi\_y0(x, t) {
18. **return** 0;
19. }
21. **function** phi\_yl(x, t) {
22. **return** x\*Math.cos(t);
23. }
25. **function** phi\_t0(x, y) {
26. **return** x\*y;
27. }
29. **function** f(x, y, t) {
30. **return** -1 \* x \* y \* Math.sin(t);
31. }

34. **function** U(x, y, t) {
35. **return** x\*y\*Math.cos(t);
36. } *//Аналитическое решение*

39. *//Параметры сетки*
40. **var** lx = 1; *//интервал x*
41. **var** Nx = 20; *//количество шагов x*
42. **var** Ny = 20; *//количество шагов y*
43. **var** K = 20; *//количество тиков*
44. **var** ly = 1; *//интервал y*
45. **var** T = 1; *//длительность*
47. **var** hx = 0; **var** hy = 0; **var** h = 0;
49. **var** a = 1; **var** b = 2; **var** c = -3;
51. **var** s1\_cur = 0; *//по y*
52. **var** s2\_cur = 0; *//по x*
53. **var** s3\_cur = 0; *//по t*
55. **var** dt = 0.02; *//длительность тика*
56. **var** dxy = 0.02; *//длительность тика по кд*
57. **var** cur\_tick = 0;
58. **var** cur\_tick2 = 0;

61. **var** generation = 0; *//номер генерации*
62. **var** go1 = **false**; *//увеличивать счетчик тиков?*
63. **var** go2 = **false**; *//увеличивать счетчик тиков?*

66. **var** setupType = 1;
67. *//1 срез по yt*
68. *//2 срез по xt*
70. **var** iters = 1000;
71. *//Количество итераций*

74. **this**.addEventListener("tick", det\_ht.bind(**this**));
75. **function** det\_ht() {
76. hx = lx / Nx;
77. hy = ly / Ny;
78. h = T / K;
80. *//определяем шаги по времени и пространству*
82. **this**.h1\_text.text = "hx = " + Math.round(hx \* 100000)/100000;
83. **this**.h2\_text.text = "hy = " + Math.round(hy \* 100000)/100000;
84. **this**.tau\_text.text = "τ = " + Math.round(h \* 100000)/100000;
86. **this**.l1\_text.text = "lx = " + Math.round(lx \* 100000)/100000;
87. **this**.l2\_text.text = "ly = " + Math.round(ly \* 100000)/100000;
88. **this**.T\_text.text = "T = " + Math.round(T \* 100000)/100000;
90. **this**.N1\_text.text = "Nx = " + Math.round(Nx);
91. **this**.N2\_text.text = "Ny = " + Math.round(Ny);
92. **this**.K\_text.text = "K = " + Math.round(K);
94. **this**.dt\_text.text = "dt = " + Math.round(dt\*1000)/1000;
95. **this**.dxy\_text.text = "dx, dy = " + Math.round(dxy\*1000)/1000;

98. **if** (setupType == 1) {
99. **if** (go1) {
100. cur\_tick += 1/60;
101. **if** (cur\_tick >= dxy && s1\_cur < Ny) {
102. **var** tGot = Math.floor(cur\_tick / dxy);
103. cur\_tick %= dxy;
104. s1\_cur += Math.min(Ny - s1\_cur, tGot);
105. }
106. **else** **if** (s1\_cur == Ny) {
107. go1 = **false**;
108. }
109. }
110. **if** (go2) {
111. cur\_tick2 += 1/60;
112. **if** (cur\_tick2 >= dt && s3\_cur < K) {
113. **var** tGot = Math.floor(cur\_tick2 / dt);
114. cur\_tick2 %= dt;
115. s3\_cur += Math.min(K - s3\_cur, tGot);
116. }
117. **else** **if** (s3\_cur == K) {
118. go2 = **false**;
119. }
120. }
121. }
122. **else** **if** (setupType == 2) {
123. **if** (go1) {
124. cur\_tick += 1/60;
125. **if** (cur\_tick >= dxy && s2\_cur < Nx) {
126. **var** tGot = Math.floor(cur\_tick / dxy);
127. cur\_tick %= dxy;
128. s2\_cur += Math.min(Nx - s2\_cur, tGot);
129. }
130. **else** **if** (s2\_cur == Nx) {
131. go1 = **false**;
132. }
133. }
134. **if** (go2) {
135. cur\_tick2 += 1/60;
136. **if** (cur\_tick2 >= dt && s3\_cur < K) {
137. **var** tGot = Math.floor(cur\_tick2 / dt);
138. cur\_tick2 %= dt;
139. s3\_cur += Math.min(K - s3\_cur, tGot);
140. }
141. **else** **if** (s3\_cur == K) {
142. go2 = **false**;
143. }
144. }
145. }
146. **else** **if** (setupType == 3) {
147. **if** (go1) {
148. cur\_tick += 1/60;
149. **if** (cur\_tick >= dxy && s2\_cur < Nx) {
150. **var** tGot = Math.floor(cur\_tick / dxy);
151. cur\_tick %= dxy;
152. s2\_cur += Math.min(Nx - s2\_cur, tGot);
153. }
154. **else** **if** (s2\_cur == Nx) {
155. go1 = **false**;
156. }
157. }
158. **if** (go2) {
159. cur\_tick2 += 1/60;
160. **if** (cur\_tick2 >= dxy && s1\_cur < Ny) {
161. **var** tGot = Math.floor(cur\_tick2 / dxy);
162. cur\_tick %= dxy;
163. s1\_cur += Math.min(Ny - s1\_cur, tGot);
164. }
165. **else** **if** (s1\_cur == Ny) {
166. go2 = **false**;
167. }
168. }
169. }
171. }



176. **this**.setGreen1.addEventListener("click", set\_l1\_prompt.bind(**this**));
177. **function** set\_l1\_prompt() {
178. generation += 1;
179. **var** temp = prompt("Введите lx:", '');
180. temp = Number.parseFloat(temp);
181. **if** (isNaN(temp)) {
182. *//если ввели какую-то хрень*
183. temp = lx;
184. }
185. **else** **if** (temp > 1000000) {
186. *//ограничение на величину*
187. temp = 1000000;
188. }
189. **else** **if** (temp <= 0) {
190. *//всегда положительно*
191. temp = 0.001;
192. }
193. lx = temp;
194. }


198. **this**.setGreen2.addEventListener("click", set\_l2\_prompt.bind(**this**));
199. **function** set\_l2\_prompt() {
200. generation += 1;
201. **var** temp = prompt("Введите ly:", '');
202. temp = Number.parseFloat(temp);
203. **if** (isNaN(temp)) {
204. *//если ввели какую-то хрень*
205. temp = ly;
206. }
207. **else** **if** (temp > 1000000) {
208. *//ограничение на величину*
209. temp = 1000000;
210. }
211. **else** **if** (temp <= 0) {
212. *//всегда положительно*
213. temp = 0.001;
214. }
215. ly = temp;
216. }


220. **this**.setGreen3.addEventListener("click", set\_T\_prompt.bind(**this**));
221. **function** set\_T\_prompt() {
222. generation += 1;
223. **var** temp = prompt("Введите T:", '');
224. temp = Number.parseFloat(temp);
225. **if** (isNaN(temp)) {
226. *//если ввели какую-то хрень*
227. temp = T;
228. }
229. **else** **if** (temp > 1000000) {
230. *//ограничение на величину*
231. temp = 1000000;
232. }
233. **else** **if** (temp <= 0) {
234. *//всегда положительно*
235. temp = 0.001;
236. }
237. T = temp;
238. }

241. **this**.setBlue1.addEventListener("click", set\_Nx\_prompt.bind(**this**));
242. **function** set\_Nx\_prompt() {
243. generation += 1;
244. **var** temp = prompt("Введите Nx:", '');
245. temp = Number.parseInt(temp);
246. **if** (isNaN(temp)) {
247. *//если ввели какую-то хрень*
248. temp = Nx;
249. }
250. **else** **if** (temp > 100) {
251. *//ограничение на величину*
252. temp = 100;
253. }
254. **else** **if** (temp <= 0) {
255. *//всегда положительно*
256. temp = 1;
257. }
258. Nx = temp;
259. }

262. **this**.setBlue2.addEventListener("click", set\_Ny\_prompt.bind(**this**));
263. **function** set\_Ny\_prompt() {
264. generation += 1;
265. **var** temp = prompt("Введите Ny:", '');
266. temp = Number.parseInt(temp);
267. **if** (isNaN(temp)) {
268. *//если ввели какую-то хрень*
269. temp = Ny;
270. }
271. **else** **if** (temp > 100) {
272. *//ограничение на величину*
273. temp = 100;
274. }
275. **else** **if** (temp <= 0) {
276. *//всегда положительно*
277. temp = 1;
278. }
279. Ny = temp;
280. }

283. **this**.setBlue3.addEventListener("click", set\_K\_prompt.bind(**this**));
284. **function** set\_K\_prompt() {
285. generation += 1;
286. **var** temp = prompt("Введите K:", '');
287. temp = Number.parseInt(temp);
288. **if** (isNaN(temp)) {
289. *//если ввели какую-то хрень*
290. temp = K;
291. }
292. **else** **if** (temp > 100) {
293. *//ограничение на величину*
294. temp = 100;
295. }
296. **else** **if** (temp <= 0) {
297. *//всегда положительно*
298. temp = 1;
299. }
300. K = temp;
301. }

304. **this**.setPurp1.addEventListener("click", set\_dt\_prompt.bind(**this**));
305. **function** set\_dt\_prompt() {
306. cur\_tick = 0;
307. **var** temp = prompt("Введите dt (для ручного переключения можно ввести очень большим):", '');
308. temp = Number.parseFloat(temp);
309. **if** (isNaN(temp)) {
310. *//если ввели какую-то хрень*
311. temp = dt;
312. }
313. **else** **if** (temp > 1000000) {
314. *//ограничение на величину*
315. temp = 1000000;
316. }
317. **else** **if** (temp < 0.02) {
318. *//всегда положительно и больше 0.02 (почти плавная смена графика)*
319. temp = 0.02;
320. }
321. dt = temp;
322. }


326. **this**.setPurp2.addEventListener("click", set\_dxy\_prompt.bind(**this**));
327. **function** set\_dxy\_prompt() {
328. cur\_tick = 0;
329. **var** temp = prompt("Введите dx и dy (для ручного переключения можно ввести очень большим):", '');
330. temp = Number.parseFloat(temp);
331. **if** (isNaN(temp)) {
332. *//если ввели какую-то хрень*
333. temp = dxy;
334. }
335. **else** **if** (temp > 1000000) {
336. *//ограничение на величину*
337. temp = 1000000;
338. }
339. **else** **if** (temp < 0.02) {
340. *//всегда положительно и больше 0.02 (почти плавная смена графика)*
341. temp = 0.02;
342. }
343. dxy = temp;
344. }





351. **var** meth\_type = 1;
352. *//Метод решения (схема)*
353. *//1 - метод переменных направлений*
354. *//2 - метод дробных шагов*

357. **var** pogr\_type = 1;
358. *//Тип погрешности*
359. *//1 - Абсолютная*
360. *//2 - От времени 1*
361. *//3 - От времени 2*
363. **var** scale\_type = 1;
364. *//Масштаб графика*
365. *//1 - Фиксированный*
366. *//2 - Динамический*
368. **this**.SLeft1.addEventListener("click", scheme\_left.bind(**this**));
369. **function** scheme\_left() {
370. **if** (setupType == 1) {
371. setupType = 3;
372. }
373. **else** {
374. setupType -= 1;
375. }
376. }
378. **this**.SRight1.addEventListener("click", scheme\_right.bind(**this**));
379. **function** scheme\_right() {
380. **if** (setupType == 3) {
381. setupType = 1;
382. }
383. **else** {
384. setupType += 1;
385. }
386. }

389. **this**.SLeft2.addEventListener("click", meth\_left.bind(**this**));
390. **function** meth\_left() {
391. **if** (meth\_type == 1) {
392. meth\_type = 2;
393. }
394. **else** {
395. meth\_type -= 1;
396. }
397. }
399. **this**.SRight2.addEventListener("click", meth\_right.bind(**this**));
400. **function** meth\_right() {
401. **if** (meth\_type == 2) {
402. meth\_type = 1;
403. }
404. **else** {
405. meth\_type += 1;
406. }
407. }


411. **this**.SLeft3.addEventListener("click", curStep\_left.bind(**this**));
412. **function** curStep\_left() {
413. **if** (setupType <= 2) {
414. **if** (s3\_cur > 0) {
415. s3\_cur -= 1;
416. }
417. }
418. **else** {
419. **if** (s2\_cur > 0) {
420. s2\_cur -= 1;
421. }
422. }
423. }


427. **this**.SLeft31.addEventListener("click", curStep\_left2.bind(**this**));
428. **function** curStep\_left2() {
429. **if** (setupType == 1 || setupType == 3) {
430. **if** (s1\_cur > 0) {
431. s1\_cur -= 1;
432. }
433. }
434. **else** {
435. **if** (s2\_cur > 0) {
436. s2\_cur -= 1;
437. }
438. }
439. }
441. **this**.SRight3.addEventListener("click", curStep\_right.bind(**this**));
442. **function** curStep\_right() {
443. **if** (setupType <= 2) {
444. **if** (s3\_cur < K) {
445. s3\_cur += 1;
446. }
447. }
448. **else** {
449. **if** (s2\_cur < Nx) {
450. s2\_cur += 1;
451. }
452. }
453. }
455. **this**.SRight31.addEventListener("click", curStep\_right2.bind(**this**));
456. **function** curStep\_right2() {
457. **if** (setupType == 1 || setupType == 3) {
458. **if** (s1\_cur < Ny) {
459. s1\_cur += 1;
460. }
461. }
462. **else** {
463. **if** (s2\_cur < Nx) {
464. s2\_cur += 1;
465. }
466. }
467. }

470. **this**.stopBtn.addEventListener("click", curStep\_stop.bind(**this**));
471. **function** curStep\_stop() {
472. dt = 1000000;
473. }
475. **this**.SLeft4.addEventListener("click", pogr\_left.bind(**this**));
476. **function** pogr\_left() {
477. **if** (pogr\_type == 1) {
478. pogr\_type = 3;
479. }
480. **else** {
481. pogr\_type -= 1;
482. }
483. }
485. **this**.SRight4.addEventListener("click", pogr\_right.bind(**this**));
486. **function** pogr\_right() {
487. **if** (pogr\_type == 3) {
488. pogr\_type = 1;
489. }
490. **else** {
491. pogr\_type += 1;
492. }
493. }
495. **this**.SLeft5.addEventListener("click", scale\_left.bind(**this**));
496. **function** scale\_left() {
497. **if** (scale\_type == 1) {
498. scale\_type = 2;
499. }
500. **else** {
501. scale\_type -= 1;
502. }
503. }
505. **this**.SRight5.addEventListener("click", scale\_right.bind(**this**));
506. **function** scale\_right() {
507. **if** (scale\_type == 2) {
508. scale\_type = 1;
509. }
510. **else** {
511. scale\_type += 1;
512. }
513. }
515. **this**.addEventListener("tick", setTexts2.bind(**this**));
516. **function** setTexts2() {
517. **if** (setupType == 1) {
518. **this**.scheme\_text.text = "Срез по yt";
519. }
520. **else** **if** (setupType == 2) {
521. **this**.scheme\_text.text = "Срез по xt";
522. }
523. **else** **if** (setupType == 3) {
524. **this**.scheme\_text.text = "Срез по xy";
525. }
527. **if** (meth\_type == 1) {
528. **this**.meth\_text.text = "Метод переменных направлений";
529. }
530. **else** **if** (meth\_type == 2) {
531. **this**.meth\_text.text = "Метод дробных шагов";
532. }

535. **if** (pogr\_type == 1) {
536. **this**.pogr\_text.text = "Абсолютная погрешность";
537. }
538. **else** **if** (pogr\_type == 2) {
539. **if** (setupType <= 2) {
540. **this**.pogr\_text.text = "Макс. Погрешность от времени";
541. }
542. **else** {
543. **this**.pogr\_text.text = "Макс. Погрешность от X";
544. }
545. }
546. **else** **if** (pogr\_type == 3) {
547. **if** (setupType == 1 || setupType == 3) {
548. **this**.pogr\_text.text = "Макс. Погрешность от Y";
549. }
550. **else** {
551. **this**.pogr\_text.text = "Макс. Погрешность от X";
552. }
553. }

556. **if** (scale\_type == 1) {
557. **this**.scale\_text.text = "Макс. Фиксированный";
558. }
559. **else** **if** (scale\_type == 2) {
560. **this**.scale\_text.text = "Динамический";
561. }

564. **if** (setupType == 1) {
565. **this**.divDown0.text = "" + 0;
566. **this**.divDown1.text = "" + Math.round(lx\*1/6\*1000)/1000;
567. **this**.divDown2.text = "" + Math.round(lx\*2/6\*1000)/1000;
568. **this**.divDown3.text = "" + Math.round(lx\*3/6\*1000)/1000;
569. **this**.divDown4.text = "" + Math.round(lx\*4/6\*1000)/1000;
570. **this**.divDown5.text = "" + Math.round(lx\*5/6\*1000)/1000;
571. **this**.divDown6.text = "" + Math.round(lx\*1000)/1000;
572. }
573. **else** **if** (setupType == 2) {
574. **this**.divDown0.text = "" + 0;
575. **this**.divDown1.text = "" + Math.round(ly\*1/6\*1000)/1000;
576. **this**.divDown2.text = "" + Math.round(ly\*2/6\*1000)/1000;
577. **this**.divDown3.text = "" + Math.round(ly\*3/6\*1000)/1000;
578. **this**.divDown4.text = "" + Math.round(ly\*4/6\*1000)/1000;
579. **this**.divDown5.text = "" + Math.round(ly\*5/6\*1000)/1000;
580. **this**.divDown6.text = "" + Math.round(ly\*1000)/1000;
581. }
582. **else** **if** (setupType == 3) {
583. **this**.divDown0.text = "" + 0;
584. **this**.divDown1.text = "" + Math.round(T\*1/6\*1000)/1000;
585. **this**.divDown2.text = "" + Math.round(T\*2/6\*1000)/1000;
586. **this**.divDown3.text = "" + Math.round(T\*3/6\*1000)/1000;
587. **this**.divDown4.text = "" + Math.round(T\*4/6\*1000)/1000;
588. **this**.divDown5.text = "" + Math.round(T\*5/6\*1000)/1000;
589. **this**.divDown6.text = "" + Math.round(T\*1000)/1000;
590. }
592. **if** (setupType == 1) {
593. **if** (pogr\_type == 1) {
594. **this**.divDown0err.text = "" + 0;
595. **this**.divDown1err.text = "" + Math.round(lx\*1/6\*1000)/1000;
596. **this**.divDown2err.text = "" + Math.round(lx\*2/6\*1000)/1000;
597. **this**.divDown3err.text = "" + Math.round(lx\*3/6\*1000)/1000;
598. **this**.divDown4err.text = "" + Math.round(lx\*4/6\*1000)/1000;
599. **this**.divDown5err.text = "" + Math.round(lx\*5/6\*1000)/1000;
600. **this**.divDown6err.text = "" + Math.round(lx\*1000)/1000;
601. }
602. **else** **if** (pogr\_type == 2) {
603. **this**.divDown0err.text = "" + 0;
604. **this**.divDown1err.text = "" + Math.round(T\*1/6\*1000)/1000;
605. **this**.divDown2err.text = "" + Math.round(T\*2/6\*1000)/1000;
606. **this**.divDown3err.text = "" + Math.round(T\*3/6\*1000)/1000;
607. **this**.divDown4err.text = "" + Math.round(T\*4/6\*1000)/1000;
608. **this**.divDown5err.text = "" + Math.round(T\*5/6\*1000)/1000;
609. **this**.divDown6err.text = "" + Math.round(T\*1000)/1000;
610. }
611. **else** **if** (pogr\_type == 3) {
612. **this**.divDown0err.text = "" + 0;
613. **this**.divDown1err.text = "" + Math.round(ly\*1/6\*1000)/1000;
614. **this**.divDown2err.text = "" + Math.round(ly\*2/6\*1000)/1000;
615. **this**.divDown3err.text = "" + Math.round(ly\*3/6\*1000)/1000;
616. **this**.divDown4err.text = "" + Math.round(ly\*4/6\*1000)/1000;
617. **this**.divDown5err.text = "" + Math.round(ly\*5/6\*1000)/1000;
618. **this**.divDown6err.text = "" + Math.round(ly\*1000)/1000;
619. }
620. }
621. **else** **if** (setupType == 2) {
622. **if** (pogr\_type == 1) {
623. **this**.divDown0err.text = "" + 0;
624. **this**.divDown1err.text = "" + Math.round(ly\*1/6\*1000)/1000;
625. **this**.divDown2err.text = "" + Math.round(ly\*2/6\*1000)/1000;
626. **this**.divDown3err.text = "" + Math.round(ly\*3/6\*1000)/1000;
627. **this**.divDown4err.text = "" + Math.round(ly\*4/6\*1000)/1000;
628. **this**.divDown5err.text = "" + Math.round(ly\*5/6\*1000)/1000;
629. **this**.divDown6err.text = "" + Math.round(ly\*1000)/1000;
630. }
631. **else** **if** (pogr\_type == 2) {
632. **this**.divDown0err.text = "" + 0;
633. **this**.divDown1err.text = "" + Math.round(T\*1/6\*1000)/1000;
634. **this**.divDown2err.text = "" + Math.round(T\*2/6\*1000)/1000;
635. **this**.divDown3err.text = "" + Math.round(T\*3/6\*1000)/1000;
636. **this**.divDown4err.text = "" + Math.round(T\*4/6\*1000)/1000;
637. **this**.divDown5err.text = "" + Math.round(T\*5/6\*1000)/1000;
638. **this**.divDown6err.text = "" + Math.round(T\*1000)/1000;
639. }
640. **else** **if** (pogr\_type == 3) {
641. **this**.divDown0err.text = "" + 0;
642. **this**.divDown1err.text = "" + Math.round(lx\*1/6\*1000)/1000;
643. **this**.divDown2err.text = "" + Math.round(lx\*2/6\*1000)/1000;
644. **this**.divDown3err.text = "" + Math.round(lx\*3/6\*1000)/1000;
645. **this**.divDown4err.text = "" + Math.round(lx\*4/6\*1000)/1000;
646. **this**.divDown5err.text = "" + Math.round(lx\*5/6\*1000)/1000;
647. **this**.divDown6err.text = "" + Math.round(lx\*1000)/1000;
648. }
649. }
650. **else** {
651. **if** (pogr\_type == 1) {
652. **this**.divDown0err.text = "" + 0;
653. **this**.divDown1err.text = "" + Math.round(T\*1/6\*1000)/1000;
654. **this**.divDown2err.text = "" + Math.round(T\*2/6\*1000)/1000;
655. **this**.divDown3err.text = "" + Math.round(T\*3/6\*1000)/1000;
656. **this**.divDown4err.text = "" + Math.round(T\*4/6\*1000)/1000;
657. **this**.divDown5err.text = "" + Math.round(T\*5/6\*1000)/1000;
658. **this**.divDown6err.text = "" + Math.round(T\*1000)/1000;
659. }
660. **else** **if** (pogr\_type == 2) {
661. **this**.divDown0err.text = "" + 0;
662. **this**.divDown1err.text = "" + Math.round(lx\*1/6\*1000)/1000;
663. **this**.divDown2err.text = "" + Math.round(lx\*2/6\*1000)/1000;
664. **this**.divDown3err.text = "" + Math.round(lx\*3/6\*1000)/1000;
665. **this**.divDown4err.text = "" + Math.round(lx\*4/6\*1000)/1000;
666. **this**.divDown5err.text = "" + Math.round(lx\*5/6\*1000)/1000;
667. **this**.divDown6err.text = "" + Math.round(lx\*1000)/1000;
668. }
669. **else** **if** (pogr\_type == 3) {
670. **this**.divDown0err.text = "" + 0;
671. **this**.divDown1err.text = "" + Math.round(ly\*1/6\*1000)/1000;
672. **this**.divDown2err.text = "" + Math.round(ly\*2/6\*1000)/1000;
673. **this**.divDown3err.text = "" + Math.round(ly\*3/6\*1000)/1000;
674. **this**.divDown4err.text = "" + Math.round(ly\*4/6\*1000)/1000;
675. **this**.divDown5err.text = "" + Math.round(ly\*5/6\*1000)/1000;
676. **this**.divDown6err.text = "" + Math.round(ly\*1000)/1000;
677. }
678. }
680. **if** (setupType == 1) {
681. **if** (maxValsxt.length > 0 && scale\_type == 2) {
682. **this**.divUp0.text = "" + Math.round(minValsyt[s3\_cur][s1\_cur]\*1000)/1000;
683. **this**.divUp1.text = "" + Math.round((minValsyt[s3\_cur][s1\_cur] + (maxValsyt[s3\_cur][s1\_cur] - minValsyt[s3\_cur][s1\_cur])\*1/7)\*1000)/1000;
684. **this**.divUp2.text = "" + Math.round((minValsyt[s3\_cur][s1\_cur] + (maxValsyt[s3\_cur][s1\_cur] - minValsyt[s3\_cur][s1\_cur])\*2/7)\*1000)/1000;
685. **this**.divUp3.text = "" + Math.round((minValsyt[s3\_cur][s1\_cur] + (maxValsyt[s3\_cur][s1\_cur] - minValsyt[s3\_cur][s1\_cur])\*3/7)\*1000)/1000;
686. **this**.divUp4.text = "" + Math.round((minValsyt[s3\_cur][s1\_cur] + (maxValsyt[s3\_cur][s1\_cur] - minValsyt[s3\_cur][s1\_cur])\*4/7)\*1000)/1000;
687. **this**.divUp5.text = "" + Math.round((minValsyt[s3\_cur][s1\_cur] + (maxValsyt[s3\_cur][s1\_cur] - minValsyt[s3\_cur][s1\_cur])\*5/7)\*1000)/1000;
688. **this**.divUp6.text = "" + Math.round((minValsyt[s3\_cur][s1\_cur] + (maxValsyt[s3\_cur][s1\_cur] - minValsyt[s3\_cur][s1\_cur])\*6/7)\*1000)/1000;
689. **this**.divUp7.text = "" + Math.round(maxValsyt[s3\_cur][s1\_cur]\*1000)/1000;
690. }
691. **else** {
692. **this**.divUp0.text = "" + Math.round(minMinValyt\*1000)/1000;
693. **this**.divUp1.text = "" + Math.round((minMinValyt + (maxMaxValyt - minMinValyt)\*1/7)\*1000)/1000;
694. **this**.divUp2.text = "" + Math.round((minMinValyt + (maxMaxValyt - minMinValyt)\*2/7)\*1000)/1000;
695. **this**.divUp3.text = "" + Math.round((minMinValyt + (maxMaxValyt - minMinValyt)\*3/7)\*1000)/1000;
696. **this**.divUp4.text = "" + Math.round((minMinValyt + (maxMaxValyt - minMinValyt)\*4/7)\*1000)/1000;
697. **this**.divUp5.text = "" + Math.round((minMinValyt + (maxMaxValyt - minMinValyt)\*5/7)\*1000)/1000;
698. **this**.divUp6.text = "" + Math.round((minMinValyt + (maxMaxValyt - minMinValyt)\*6/7)\*1000)/1000;
699. **this**.divUp7.text = "" + Math.round(maxMaxValyt\*1000)/1000;
700. }
701. }
702. **else** **if** (setupType == 2) {
703. **if** (maxValsyt.length > 0 && scale\_type == 2) {
704. **this**.divUp0.text = "" + Math.round(minValsxt[s3\_cur][s2\_cur]\*1000)/1000;
705. **this**.divUp1.text = "" + Math.round((minValsxt[s3\_cur][s2\_cur] + (maxValsxt[s3\_cur][s2\_cur] - minValsxt[s3\_cur][s2\_cur])\*1/7)\*1000)/1000;
706. **this**.divUp2.text = "" + Math.round((minValsxt[s3\_cur][s2\_cur] + (maxValsxt[s3\_cur][s2\_cur] - minValsxt[s3\_cur][s2\_cur])\*2/7)\*1000)/1000;
707. **this**.divUp3.text = "" + Math.round((minValsxt[s3\_cur][s2\_cur] + (maxValsxt[s3\_cur][s2\_cur] - minValsxt[s3\_cur][s2\_cur])\*3/7)\*1000)/1000;
708. **this**.divUp4.text = "" + Math.round((minValsxt[s3\_cur][s2\_cur] + (maxValsxt[s3\_cur][s2\_cur] - minValsxt[s3\_cur][s2\_cur])\*4/7)\*1000)/1000;
709. **this**.divUp5.text = "" + Math.round((minValsxt[s3\_cur][s2\_cur] + (maxValsxt[s3\_cur][s2\_cur] - minValsxt[s3\_cur][s2\_cur])\*5/7)\*1000)/1000;
710. **this**.divUp6.text = "" + Math.round((minValsxt[s3\_cur][s2\_cur] + (maxValsxt[s3\_cur][s2\_cur] - minValsxt[s3\_cur][s2\_cur])\*6/7)\*1000)/1000;
711. **this**.divUp7.text = "" + Math.round(maxValsxt[s3\_cur][s2\_cur]\*1000)/1000;
712. }
713. **else** {
714. **this**.divUp0.text = "" + Math.round(minMinValxt\*1000)/1000;
715. **this**.divUp1.text = "" + Math.round((minMinValxt + (maxMaxValxt - minMinValxt)\*1/7)\*1000)/1000;
716. **this**.divUp2.text = "" + Math.round((minMinValxt + (maxMaxValxt - minMinValxt)\*2/7)\*1000)/1000;
717. **this**.divUp3.text = "" + Math.round((minMinValxt + (maxMaxValxt - minMinValxt)\*3/7)\*1000)/1000;
718. **this**.divUp4.text = "" + Math.round((minMinValxt + (maxMaxValxt - minMinValxt)\*4/7)\*1000)/1000;
719. **this**.divUp5.text = "" + Math.round((minMinValxt + (maxMaxValxt - minMinValxt)\*5/7)\*1000)/1000;
720. **this**.divUp6.text = "" + Math.round((minMinValxt + (maxMaxValxt - minMinValxt)\*6/7)\*1000)/1000;
721. **this**.divUp7.text = "" + Math.round(maxMaxValxt\*1000)/1000;
722. }
723. }
724. **else** {
725. **if** (maxValsxy.length > 0 && scale\_type == 2) {
726. **this**.divUp0.text = "" + Math.round(minValsxy[s1\_cur][s2\_cur]\*1000)/1000;
727. **this**.divUp1.text = "" + Math.round((minValsxy[s1\_cur][s2\_cur] + (maxValsxy[s1\_cur][s2\_cur] - minValsxy[s1\_cur][s2\_cur])\*1/7)\*1000)/1000;
728. **this**.divUp2.text = "" + Math.round((minValsxy[s1\_cur][s2\_cur] + (maxValsxy[s1\_cur][s2\_cur] - minValsxy[s1\_cur][s2\_cur])\*2/7)\*1000)/1000;
729. **this**.divUp3.text = "" + Math.round((minValsxy[s1\_cur][s2\_cur] + (maxValsxy[s1\_cur][s2\_cur] - minValsxy[s1\_cur][s2\_cur])\*3/7)\*1000)/1000;
730. **this**.divUp4.text = "" + Math.round((minValsxy[s1\_cur][s2\_cur] + (maxValsxy[s1\_cur][s2\_cur] - minValsxy[s1\_cur][s2\_cur])\*4/7)\*1000)/1000;
731. **this**.divUp5.text = "" + Math.round((minValsxy[s1\_cur][s2\_cur] + (maxValsxy[s1\_cur][s2\_cur] - minValsxy[s1\_cur][s2\_cur])\*5/7)\*1000)/1000;
732. **this**.divUp6.text = "" + Math.round((minValsxy[s1\_cur][s2\_cur] + (maxValsxy[s1\_cur][s2\_cur] - minValsxy[s1\_cur][s2\_cur])\*6/7)\*1000)/1000;
733. **this**.divUp7.text = "" + Math.round(maxValsxy[s1\_cur][s2\_cur]\*1000)/1000;
734. }
735. **else** {
736. **this**.divUp0.text = "" + Math.round(minMinValxy\*1000)/1000;
737. **this**.divUp1.text = "" + Math.round((minMinValxy + (maxMaxValxy - minMinValxy)\*1/7)\*1000)/1000;
738. **this**.divUp2.text = "" + Math.round((minMinValxy + (maxMaxValxy - minMinValxy)\*2/7)\*1000)/1000;
739. **this**.divUp3.text = "" + Math.round((minMinValxy + (maxMaxValxy - minMinValxy)\*3/7)\*1000)/1000;
740. **this**.divUp4.text = "" + Math.round((minMinValxy + (maxMaxValxy - minMinValxy)\*4/7)\*1000)/1000;
741. **this**.divUp5.text = "" + Math.round((minMinValxy + (maxMaxValxy - minMinValxy)\*5/7)\*1000)/1000;
742. **this**.divUp6.text = "" + Math.round((minMinValxy + (maxMaxValxy - minMinValxy)\*6/7)\*1000)/1000;
743. **this**.divUp7.text = "" + Math.round(maxMaxValxy\*1000)/1000;
744. }
745. }
747. **if** (setupType == 1) {
748. **if** (maxValsE.length > 0 && scale\_type == 2 && pogr\_type == 1) {
749. **this**.divUp0err.text = "" + Math.round(minValsE[s3\_cur][s1\_cur]\*1000)/1000;
750. **this**.divUp1err.text = "" + Math.round((minValsE[s3\_cur][s1\_cur] + (maxValsE[s3\_cur][s1\_cur] - minValsE[s3\_cur][s1\_cur])\*1/7)\*1000)/1000;
751. **this**.divUp2err.text = "" + Math.round((minValsE[s3\_cur][s1\_cur] + (maxValsE[s3\_cur][s1\_cur] - minValsE[s3\_cur][s1\_cur])\*2/7)\*1000)/1000;
752. **this**.divUp3err.text = "" + Math.round((minValsE[s3\_cur][s1\_cur] + (maxValsE[s3\_cur][s1\_cur] - minValsE[s3\_cur][s1\_cur])\*3/7)\*1000)/1000;
753. **this**.divUp4err.text = "" + Math.round((minValsE[s3\_cur][s1\_cur] + (maxValsE[s3\_cur][s1\_cur] - minValsE[s3\_cur][s1\_cur])\*4/7)\*1000)/1000;
754. **this**.divUp5err.text = "" + Math.round((minValsE[s3\_cur][s1\_cur] + (maxValsE[s3\_cur][s1\_cur] - minValsE[s3\_cur][s1\_cur])\*5/7)\*1000)/1000;
755. **this**.divUp6err.text = "" + Math.round((minValsE[s3\_cur][s1\_cur] + (maxValsE[s3\_cur][s1\_cur] - minValsE[s3\_cur][s1\_cur])\*6/7)\*1000)/1000;
756. **this**.divUp7err.text = "" + Math.round(maxValsE[s3\_cur][s1\_cur]\*1000)/1000;
757. }
758. **else** **if** (pogr\_type == 1) {
759. **this**.divUp0err.text = "" + Math.round(minMinValE\*1000)/1000;
760. **this**.divUp1err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*1/7)\*1000)/1000;
761. **this**.divUp2err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*2/7)\*1000)/1000;
762. **this**.divUp3err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*3/7)\*1000)/1000;
763. **this**.divUp4err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*4/7)\*1000)/1000;
764. **this**.divUp5err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*5/7)\*1000)/1000;
765. **this**.divUp6err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*6/7)\*1000)/1000;
766. **this**.divUp7err.text = "" + Math.round(maxMaxValE\*1000)/1000;
767. }
768. **else** **if** (pogr\_type == 2) {
769. **this**.divUp0err.text = "" + Math.round(minEotTt\*10000)/10000;
770. **this**.divUp1err.text = "" + Math.round((minEotTt + (maxEotTt - minEotTt)\*1/7)\*10000)/10000;
771. **this**.divUp2err.text = "" + Math.round((minEotTt + (maxEotTt - minEotTt)\*2/7)\*10000)/10000;
772. **this**.divUp3err.text = "" + Math.round((minEotTt + (maxEotTt - minEotTt)\*3/7)\*10000)/10000;
773. **this**.divUp4err.text = "" + Math.round((minEotTt + (maxEotTt - minEotTt)\*4/7)\*10000)/10000;
774. **this**.divUp5err.text = "" + Math.round((minEotTt + (maxEotTt - minEotTt)\*5/7)\*10000)/10000;
775. **this**.divUp6err.text = "" + Math.round((minEotTt + (maxEotTt - minEotTt)\*6/7)\*10000)/10000;
776. **this**.divUp7err.text = "" + Math.round(maxEotTt\*10000)/10000;
777. }
778. **else** **if** (pogr\_type == 3) {
779. **this**.divUp0err.text = "" + Math.round(minEotTy\*10000)/10000;
780. **this**.divUp1err.text = "" + Math.round((minEotTy + (maxEotTy - minEotTy)\*1/7)\*10000)/10000;
781. **this**.divUp2err.text = "" + Math.round((minEotTy + (maxEotTy - minEotTy)\*2/7)\*10000)/10000;
782. **this**.divUp3err.text = "" + Math.round((minEotTy + (maxEotTy - minEotTy)\*3/7)\*10000)/10000;
783. **this**.divUp4err.text = "" + Math.round((minEotTy + (maxEotTy - minEotTy)\*4/7)\*10000)/10000;
784. **this**.divUp5err.text = "" + Math.round((minEotTy + (maxEotTy - minEotTy)\*5/7)\*10000)/10000;
785. **this**.divUp6err.text = "" + Math.round((minEotTy + (maxEotTy - minEotTy)\*6/7)\*10000)/10000;
786. **this**.divUp7err.text = "" + Math.round(maxEotTy\*10000)/10000;
787. }
788. }
789. **else** **if** (setupType == 2) {
790. **if** (maxValsE.length > 0 && scale\_type == 2 && pogr\_type == 1) {
791. **this**.divUp0err.text = "" + Math.round(minValsE[s3\_cur][s2\_cur]\*1000)/1000;
792. **this**.divUp1err.text = "" + Math.round((minValsE[s3\_cur][s2\_cur] + (maxValsE[s3\_cur][s2\_cur] - minValsE[s3\_cur][s2\_cur])\*1/7)\*1000)/1000;
793. **this**.divUp2err.text = "" + Math.round((minValsE[s3\_cur][s2\_cur] + (maxValsE[s3\_cur][s2\_cur] - minValsE[s3\_cur][s2\_cur])\*2/7)\*1000)/1000;
794. **this**.divUp3err.text = "" + Math.round((minValsE[s3\_cur][s2\_cur] + (maxValsE[s3\_cur][s2\_cur] - minValsE[s3\_cur][s2\_cur])\*3/7)\*1000)/1000;
795. **this**.divUp4err.text = "" + Math.round((minValsE[s3\_cur][s2\_cur] + (maxValsE[s3\_cur][s2\_cur] - minValsE[s3\_cur][s2\_cur])\*4/7)\*1000)/1000;
796. **this**.divUp5err.text = "" + Math.round((minValsE[s3\_cur][s2\_cur] + (maxValsE[s3\_cur][s2\_cur] - minValsE[s3\_cur][s2\_cur])\*5/7)\*1000)/1000;
797. **this**.divUp6err.text = "" + Math.round((minValsE[s3\_cur][s2\_cur] + (maxValsE[s3\_cur][s2\_cur] - minValsE[s3\_cur][s2\_cur])\*6/7)\*1000)/1000;
798. **this**.divUp7err.text = "" + Math.round(maxValsE[s3\_cur][s2\_cur]\*1000)/1000;
799. }
800. **else** **if** (pogr\_type == 1) {
801. **this**.divUp0err.text = "" + Math.round(minMinValE\*1000)/1000;
802. **this**.divUp1err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*1/7)\*1000)/1000;
803. **this**.divUp2err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*2/7)\*1000)/1000;
804. **this**.divUp3err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*3/7)\*1000)/1000;
805. **this**.divUp4err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*4/7)\*1000)/1000;
806. **this**.divUp5err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*5/7)\*1000)/1000;
807. **this**.divUp6err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*6/7)\*1000)/1000;
808. **this**.divUp7err.text = "" + Math.round(maxMaxValE\*1000)/1000;
809. }
810. **else** **if** (pogr\_type == 2) {
811. **this**.divUp0err.text = "" + Math.round(minEotTt2\*10000)/10000;
812. **this**.divUp1err.text = "" + Math.round((minEotTt2 + (maxEotTt2 - minEotTt2)\*1/7)\*10000)/10000;
813. **this**.divUp2err.text = "" + Math.round((minEotTt2 + (maxEotTt2 - minEotTt2)\*2/7)\*10000)/10000;
814. **this**.divUp3err.text = "" + Math.round((minEotTt2 + (maxEotTt2 - minEotTt2)\*3/7)\*10000)/10000;
815. **this**.divUp4err.text = "" + Math.round((minEotTt2 + (maxEotTt2 - minEotTt2)\*4/7)\*10000)/10000;
816. **this**.divUp5err.text = "" + Math.round((minEotTt2 + (maxEotTt2 - minEotTt2)\*5/7)\*10000)/10000;
817. **this**.divUp6err.text = "" + Math.round((minEotTt2 + (maxEotTt2 - minEotTt2)\*6/7)\*10000)/10000;
818. **this**.divUp7err.text = "" + Math.round(maxEotTt2\*10000)/10000;
819. }
820. **else** **if** (pogr\_type == 3) {
821. **this**.divUp0err.text = "" + Math.round(minEotTx2\*10000)/10000;
822. **this**.divUp1err.text = "" + Math.round((minEotTx2 + (maxEotTx2 - minEotTx2)\*1/7)\*10000)/10000;
823. **this**.divUp2err.text = "" + Math.round((minEotTx2 + (maxEotTx2 - minEotTx2)\*2/7)\*10000)/10000;
824. **this**.divUp3err.text = "" + Math.round((minEotTx2 + (maxEotTx2 - minEotTx2)\*3/7)\*10000)/10000;
825. **this**.divUp4err.text = "" + Math.round((minEotTx2 + (maxEotTx2 - minEotTx2)\*4/7)\*10000)/10000;
826. **this**.divUp5err.text = "" + Math.round((minEotTx2 + (maxEotTx2 - minEotTx2)\*5/7)\*10000)/10000;
827. **this**.divUp6err.text = "" + Math.round((minEotTx2 + (maxEotTx2 - minEotTx2)\*6/7)\*10000)/10000;
828. **this**.divUp7err.text = "" + Math.round(maxEotTx2\*10000)/10000;
829. }
830. }
831. **else** {
832. **if** (maxValsE.length > 0 && scale\_type == 2 && pogr\_type == 1) {
833. **this**.divUp0err.text = "" + Math.round(minValsE[s1\_cur][s2\_cur]\*1000)/1000;
834. **this**.divUp1err.text = "" + Math.round((minValsE[s1\_cur][s2\_cur] + (maxValsE[s1\_cur][s2\_cur] - minValsE[s1\_cur][s2\_cur])\*1/7)\*1000)/1000;
835. **this**.divUp2err.text = "" + Math.round((minValsE[s1\_cur][s2\_cur] + (maxValsE[s1\_cur][s2\_cur] - minValsE[s1\_cur][s2\_cur])\*2/7)\*1000)/1000;
836. **this**.divUp3err.text = "" + Math.round((minValsE[s1\_cur][s2\_cur] + (maxValsE[s1\_cur][s2\_cur] - minValsE[s1\_cur][s2\_cur])\*3/7)\*1000)/1000;
837. **this**.divUp4err.text = "" + Math.round((minValsE[s1\_cur][s2\_cur] + (maxValsE[s1\_cur][s2\_cur] - minValsE[s1\_cur][s2\_cur])\*4/7)\*1000)/1000;
838. **this**.divUp5err.text = "" + Math.round((minValsE[s1\_cur][s2\_cur] + (maxValsE[s1\_cur][s2\_cur] - minValsE[s1\_cur][s2\_cur])\*5/7)\*1000)/1000;
839. **this**.divUp6err.text = "" + Math.round((minValsE[s1\_cur][s2\_cur] + (maxValsE[s1\_cur][s2\_cur] - minValsE[s1\_cur][s2\_cur])\*6/7)\*1000)/1000;
840. **this**.divUp7err.text = "" + Math.round(maxValsE[s1\_cur][s2\_cur]\*1000)/1000;
841. }
842. **else** **if** (pogr\_type == 1) {
843. **this**.divUp0err.text = "" + Math.round(minMinValE\*1000)/1000;
844. **this**.divUp1err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*1/7)\*1000)/1000;
845. **this**.divUp2err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*2/7)\*1000)/1000;
846. **this**.divUp3err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*3/7)\*1000)/1000;
847. **this**.divUp4err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*4/7)\*1000)/1000;
848. **this**.divUp5err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*5/7)\*1000)/1000;
849. **this**.divUp6err.text = "" + Math.round((minMinValE + (maxMaxValE - minMinValE)\*6/7)\*1000)/1000;
850. **this**.divUp7err.text = "" + Math.round(maxMaxValE\*1000)/1000;
851. }
852. **else** **if** (pogr\_type == 2) {
853. **this**.divUp0err.text = "" + Math.round(minEotTx\*10000)/10000;
854. **this**.divUp1err.text = "" + Math.round((minEotTx + (maxEotTx - minEotTx)\*1/7)\*10000)/10000;
855. **this**.divUp2err.text = "" + Math.round((minEotTx + (maxEotTx - minEotTx)\*2/7)\*10000)/10000;
856. **this**.divUp3err.text = "" + Math.round((minEotTx + (maxEotTx - minEotTx)\*3/7)\*10000)/10000;
857. **this**.divUp4err.text = "" + Math.round((minEotTx + (maxEotTx - minEotTx)\*4/7)\*10000)/10000;
858. **this**.divUp5err.text = "" + Math.round((minEotTx + (maxEotTx - minEotTx)\*5/7)\*10000)/10000;
859. **this**.divUp6err.text = "" + Math.round((minEotTx + (maxEotTx - minEotTx)\*6/7)\*10000)/10000;
860. **this**.divUp7err.text = "" + Math.round(maxEotTx\*10000)/10000;
861. }
862. **else** **if** (pogr\_type == 3) {
863. **this**.divUp0err.text = "" + Math.round(minEotTy2\*10000)/10000;
864. **this**.divUp1err.text = "" + Math.round((minEotTy2 + (maxEotTy2 - minEotTy2)\*1/7)\*10000)/10000;
865. **this**.divUp2err.text = "" + Math.round((minEotTy2 + (maxEotTy2 - minEotTy2)\*2/7)\*10000)/10000;
866. **this**.divUp3err.text = "" + Math.round((minEotTy2 + (maxEotTy2 - minEotTy2)\*3/7)\*10000)/10000;
867. **this**.divUp4err.text = "" + Math.round((minEotTy2 + (maxEotTy2 - minEotTy2)\*4/7)\*10000)/10000;
868. **this**.divUp5err.text = "" + Math.round((minEotTy2 + (maxEotTy2 - minEotTy2)\*5/7)\*10000)/10000;
869. **this**.divUp6err.text = "" + Math.round((minEotTy2 + (maxEotTy2 - minEotTy2)\*6/7)\*10000)/10000;
870. **this**.divUp7err.text = "" + Math.round(maxEotTy2\*10000)/10000;
871. }
872. }

875. **if** (setupType == 1) {
876. **var** s1\_cur2 = s1\_cur\*hy;
877. **var** s3\_cur2 = s3\_cur\*h;
878. **this**.step\_text.text = "t = " + Math.round(s3\_cur2\*10000)/10000 + " (шаг " + Math.round(s3\_cur) + ")";
879. **this**.step\_text2.text = "y = " + Math.round(s1\_cur2\*10000)/10000 + " (шаг " + Math.round(s1\_cur) + ")";
880. **this**.axisDown.axisTp.text = "X";
881. }
882. **else** **if** (setupType == 2) {
883. **var** s2\_cur2 = s2\_cur\*hx;
884. **var** s3\_cur2 = s3\_cur\*h;
885. **this**.step\_text.text = "t = " + Math.round(s3\_cur2\*10000)/10000 + " (шаг " + Math.round(s3\_cur) + ")";
886. **this**.step\_text2.text = "x = " + Math.round(s2\_cur2\*10000)/10000 + " (шаг " + Math.round(s2\_cur) + ")";
887. **this**.axisDown.axisTp.text = "Y";
888. }
889. **else** **if** (setupType == 3) {
890. **var** s1\_cur2 = s1\_cur\*hy;
891. **var** s2\_cur2 = s2\_cur\*hx;
892. **this**.step\_text.text = "x = " + Math.round(s2\_cur2\*10000)/10000 + " (шаг " + Math.round(s2\_cur) + ")";
893. **this**.step\_text2.text = "y = " + Math.round(s1\_cur2\*10000)/10000 + " (шаг " + Math.round(s1\_cur) + ")";
894. **this**.axisDown.axisTp.text = "t";
895. }
896. **this**.axisDown.y = Math.min(Math.max(transfY(0), 26.4), 418.6);

899. **this**.axisDownErr.y = Math.min(Math.max(transfEY(0), 468), 700);
900. }

903. *//baseGraph*
904. *//x = 0 : posX = 60*
905. *//y = 0 : posY = 418.6*
907. *//x = max : posX = 627*
908. *//y = max : posY = 26.4*
910. *//errGraph*
911. *//x = 0 : posX = 53.6*
912. *//y = 0 : posY = 700*
914. *//x = max : posX = 397.8*
915. *//y = max : posY = 468.4*




921. **var** s1\_cur = 0;
922. *//текущий шаг по y*
924. **var** s2\_cur = 0;
925. *//текущий шаг по x*
927. **var** s3\_cur = 0;
928. *//текущий шаг по t*

931. **var** etalon = [];
932. **var** xVector = [];
933. **var** yVector = [];
934. **var** tVector = [];
936. **var** errorCube = []
938. **var** maxMaxValyt = 1;
939. **var** minMinValyt = 0;
941. **var** maxMaxValxt = 1;
942. **var** minMinValxt = 0;
944. **var** maxMaxValxy = 1;
945. **var** minMinValxy = 0;
947. **var** maxMaxValE = 1; *//link*
948. **var** minMinValE = 0; *//link*
950. **var** maxMaxValE1yt = 1;
951. **var** minMinValE1yt = 0;
953. **var** maxMaxValE1xt = 1;
954. **var** minMinValE1xt = 0;
956. **var** maxMaxValE1xy = 1;
957. **var** minMinValE1xy = 0;
959. **var** maxValsyt = [];
960. **var** minValsyt = [];
962. **var** maxValsxt = [];
963. **var** minValsxt = [];
965. **var** maxValsxy = [];
966. **var** minValsxy = [];
968. **var** maxValsE = []; *//link*
969. **var** minValsE = []; *//link*
971. **var** maxValsE1yt = [];
972. **var** minValsE1yt = [];
974. **var** maxValsE1xt = [];
975. **var** minValsE1xt = [];
977. **var** maxValsE1xy = [];
978. **var** minValsE1xy = [];


982. **var** EotTx = [];
983. **var** EotTy = [];
984. **var** EotTt = [];
986. **var** EotTx2 = [];
987. **var** EotTy2 = [];
988. **var** EotTt2 = [];
990. **this**.addEventListener("tick", setError.bind(**this**));
991. **function** setError() {
992. **if** (setupType == 1) {
993. maxMaxValE = maxMaxValE1yt;
994. minMinValE = minMinValE1yt;
995. maxValsE = maxValsE1yt;
996. minValsE = minValsE1yt;
997. }
998. **else** **if** (setupType == 2) {
999. maxMaxValE = maxMaxValE1xt;
1000. minMinValE = minMinValE1xt;
1001. maxValsE = maxValsE1xt;
1002. minValsE = minValsE1xt;
1004. }
1005. **else** **if** (setupType == 3) {
1006. maxMaxValE = maxMaxValE1xy;
1007. minMinValE = minMinValE1xy;
1008. maxValsE = maxValsE1xy;
1009. minValsE = minValsE1xy;
1010. }
1011. }
1013. **var** solve = [];
1014. **var** solve2 = [];

1017. **function** copyV(vector) {
1018. **var** newV = [];
1019. **for** (**var** i = 0 ; i < vector.length ; ++i) {
1020. newV.push(vector[i]);
1021. }
1022. **return** newV;
1023. }
1025. **function** copyM(matrix) {
1026. **var** newM = [];
1027. **for** (**var** i = 0 ; i < matrix.length ; ++i) {
1028. newM.push(copyV(matrix[i]));
1029. }
1030. **return** newM;
1031. }
1033. **function** copyC(cube) {
1034. **var** newC = [];
1035. **for** (**var** i = 0 ; i < cube.length ; ++i) {
1036. newC.push(copyM(cube[i]));
1037. }
1038. **return** newC;
1039. }

1042. **var** minEotTx = 0;
1043. **var** maxEotTx = 1;
1045. **var** minEotTy = 0;
1046. **var** maxEotTy = 1;
1048. **var** minEotTt = 0;
1049. **var** maxEotTt = 1;
1051. **var** minEotTx2 = 0;
1052. **var** maxEotTx2 = 1;
1054. **var** minEotTy2 = 0;
1055. **var** maxEotTy2 = 1;
1057. **var** minEotTt2 = 0;
1058. **var** maxEotTt2 = 1;



1063. **this**.beginBtn.addEventListener("click", beginSimulation.bind(**this**));
1064. **function** beginSimulation() {
1065. generation += 1;
1066. s1\_cur = 0;
1067. s2\_cur = 0;
1068. s3\_cur = 0;
1069. etalon = [];
1070. solve = [];
1071. solve2 = [];
1072. maxMaxValyt = 1;
1073. minMinValyt = 0;
1075. maxMaxValxt = 1;
1076. minMinValxt = 0;
1078. maxMaxValxy = 1;
1079. minMinValxy = 0;
1081. maxValsyt = [];
1082. minValsyt = [];
1084. **for** (**var** t = 0; t <= K ; ++t) {
1085. **var** line = [];
1086. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1087. line.push(0);
1088. }
1089. maxValsyt.push(copyV(line));
1090. minValsyt.push(copyV(line));
1091. }
1093. maxValsxt = [];
1094. minValsxt = [];
1096. **for** (**var** t = 0; t <= K ; ++t) {
1097. **var** line = [];
1098. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1099. line.push(0);
1100. }
1101. maxValsxt.push(copyV(line));
1102. minValsxt.push(copyV(line));
1103. }
1105. maxValsxy = [];
1106. minValsxy = [];
1108. **for** (**var** y = 0; y <= Ny ; ++y) {
1109. **var** line = [];
1110. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1111. line.push(0);
1112. }
1113. maxValsxy.push(copyV(line));
1114. minValsxy.push(copyV(line));
1115. }
1117. xVector = [];
1118. yVector = [];
1119. tVector = [];
1121. **for** (**var** j = 0 ; j <= Nx ; ++j) {
1122. xVector.push(hx \* j);
1123. }
1124. **for** (**var** j = 0 ; j <= Ny ; ++j) {
1125. yVector.push(hy \* j);
1126. }
1127. **for** (**var** j = 0 ; j <= K ; ++j) {
1128. tVector.push(h \* j);
1129. }



1134. **for** (**var** i = 0 ; i <= K ; ++i) {
1135. **var** timeMatrix = [];
1136. **var** this\_t = tVector[i];
1138. **var** this\_maxX = 0;
1139. **var** this\_minX = 0;
1141. **for** (**var** j = 0 ; j <= Ny ; ++j) {
1142. **var** timeVector = [];
1143. **var** this\_y = yVector[j];
1145. **for** (**var** k = 0 ; k <= Nx ; ++k) {
1146. **var** this\_x = xVector[k];
1147. **var** func\_res = U(this\_x, this\_y, this\_t);
1149. **if** (k == 0) {
1150. maxValsyt[i][j] = func\_res;
1151. minValsyt[i][j] = func\_res;
1152. }
1153. **else** {
1154. maxValsyt[i][j] = (func\_res > maxValsyt[i][j] ? func\_res : maxValsyt[i][j]);
1155. minValsyt[i][j] = (func\_res < minValsyt[i][j] ? func\_res : minValsyt[i][j]);
1156. }

1159. **if** (j == 0) {
1160. maxValsxt[i][k] = func\_res;
1161. minValsxt[i][k] = func\_res;
1162. }
1163. **else** {
1164. maxValsxt[i][k] = (func\_res > maxValsxt[i][k] ? func\_res : maxValsxt[i][k]);
1165. minValsxt[i][k] = (func\_res < minValsxt[i][k] ? func\_res : minValsxt[i][k]);
1166. }

1169. **if** (i == 0) {
1170. maxValsxy[j][k] = func\_res;
1171. minValsxy[j][k] = func\_res;
1172. }
1173. **else** {
1174. maxValsxy[j][k] = (func\_res > maxValsxy[j][k] ? func\_res : maxValsxy[j][k]);
1175. minValsxy[j][k] = (func\_res < minValsxy[j][k] ? func\_res : minValsxy[j][k]);
1176. }

1179. timeVector.push(func\_res);
1180. }
1182. timeMatrix.push(timeVector);
1184. }
1185. etalon.push(timeMatrix);
1186. }
1188. solve2 = [];
1189. **var** downGrid = [];*//нижний слой*
1190. **var** emptyGrid = [];*//пустая сетка*
1192. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1193. **var** timeVector = [];
1194. **var** emptyVector = [];
1195. **var** this\_y = yVector[y];
1196. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1197. **var** this\_x = xVector[x];
1198. **var** res = phi\_t0(this\_x, this\_y);
1199. timeVector.push(res);
1200. emptyVector.push(0);
1201. }
1202. downGrid.push(timeVector);
1203. emptyGrid.push(emptyVector);
1204. }
1206. solve2.push(downGrid);
1208. *//метод переменных направлений*
1209. **if** (meth\_type == 1) {
1210. **for** (**var** it = 1 ; it <= 2\*K ; ++it) {
1211. solve2.push(copyM(emptyGrid));
1212. **var** curT = it\*h/2;
1214. *//обрабатываем граничные условия*
1215. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1216. **var** curY = y\*hy;
1217. solve2[it][y][0] = phi\_x0(curY, curT);
1218. solve2[it][y][Nx] = phi\_xl(curY, curT);
1219. }
1221. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1222. **var** curX = x\*hx;
1223. solve2[it][0][x] = phi\_y0(curX, curT);
1224. solve2[it][Ny][x] = phi\_yl(curX, curT);
1225. }
1227. **var** itsMatrix = [];
1229. *//нечетный шаг схемы переменных направлений*
1231. **if** (it%2 == 1) {
1232. **for** (**var** j = 1 ; j < Ny ; ++j) {
1233. **var** curY = j\*hy;
1234. **var** solveMatr = [];
1235. **var** resVector = [];
1236. **var** BVector = [];
1237. **for** (**var** i = 0 ; i <= Nx ; ++i) {
1238. **var** curX = i\*hx;
1239. **var** freeVector = [];
1241. **for** (**var** ii = 0 ; ii <= Nx ; ++ii) {
1242. **if** (i == 0) {
1243. **if** (ii == 0) {
1244. freeVector.push(1);
1245. }
1246. **else** {
1247. freeVector.push(0);
1248. }
1249. }
1250. **else** **if** (i < Nx) {
1251. **if** (ii < i - 1) {
1252. freeVector.push(0);
1253. }
1254. **else** **if** (ii == i - 1) {
1255. **var** A = h / (2 \* hx \* hx);
1256. freeVector.push(A);
1257. }
1258. **else** **if** (ii == i) {
1259. **var** B = -1 - (h / (hx \* hx));
1260. freeVector.push(B);
1261. }
1262. **else** **if** (ii == i + 1) {
1263. **var** C = h / (2 \* hx \* hx);
1264. freeVector.push(C);
1265. }
1266. **else** {
1267. freeVector.push(0);
1268. }
1269. }
1270. **else** {
1271. **if** (ii == Nx) {
1272. freeVector.push(1);
1273. }
1274. **else** {
1275. freeVector.push(0);
1276. }
1277. }
1278. }
1279. solveMatr.push(freeVector);
1281. **if** (i == 0) {
1282. BVector.push(phi\_x0(curY, curT));
1283. }
1284. **else** **if** (i < Nx) {
1285. **var** D = -h/(2\*hy\*hy)\*(solve2[it-1][j+1][i]
1286. - 2\*solve2[it-1][j][i] + solve2[it-1][j-1][i])
1287. - solve2[it-1][j][i] - (h/2)\*f(curX, curY, curT);

1290. BVector.push(D);
1291. }
1292. **else** {
1293. BVector.push(phi\_xl(curY, curT));
1294. }
1295. }
1297. resVector = progonka(solveMatr, BVector);
1298. itsMatrix.push(copyV(resVector));
1299. }

1302. **for** (**var** y = 0 ; y < Ny - 1 ; ++y) {
1303. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1304. solve2[it][y+1][x] = itsMatrix[y][x];
1305. }
1306. }
1307. }
1308. **else** {
1309. **for** (**var** j = 1 ; j < Nx ; ++j) {
1310. **var** curX = j\*hx;
1311. **var** solveMatr = [];
1312. **var** resVector = [];
1313. **var** BVector = [];
1314. **for** (**var** i = 0 ; i <= Ny ; ++i) {
1315. **var** curY = i\*hy;
1316. **var** freeVector = [];
1318. **for** (**var** ii = 0 ; ii <= Ny ; ++ii) {
1319. **if** (i == 0) {
1320. **if** (ii == 0) {
1321. freeVector.push(1);
1322. }
1323. **else** {
1324. freeVector.push(0);
1325. }
1326. }
1327. **else** **if** (i < Ny) {
1328. **if** (ii < i - 1) {
1329. freeVector.push(0);
1330. }
1331. **else** **if** (ii == i - 1) {
1332. **var** A = h / (2 \* hy \* hy);
1333. freeVector.push(A);
1334. }
1335. **else** **if** (ii == i) {
1336. **var** B = -1 - (h / (hy \* hy));
1337. freeVector.push(B);
1338. }
1339. **else** **if** (ii == i + 1) {
1340. **var** C = h / (2 \* hy \* hy);
1341. freeVector.push(C);
1342. }
1343. **else** {
1344. freeVector.push(0);
1345. }
1346. }
1347. **else** {
1348. **if** (ii == Ny) {
1349. freeVector.push(1);
1350. }
1351. **else** {
1352. freeVector.push(0);
1353. }
1354. }
1355. }
1356. solveMatr.push(copyV(freeVector));
1358. **if** (i == 0) {
1359. BVector.push(phi\_y0(curX, curT - h/2));
1360. }
1361. **else** **if** (i < Ny) {
1362. **var** D = -h/(2\*hx\*hx)\*(solve2[it-1][i][j+1]
1363. - 2\*solve2[it-1][i][j] + solve2[it-1][i][j-1])
1364. - solve2[it-1][i][j] - (h/2)\*f(curX, curY, curT - h/2);
1366. BVector.push(D);
1367. }
1368. **else** {
1369. BVector.push(phi\_yl(curX, curT - h/2));
1370. }
1371. }
1373. resVector = progonka(solveMatr, BVector);

1376. itsMatrix.push(copyV(resVector));
1377. }


1381. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1382. **for** (**var** x = 0 ; x < Nx - 1 ; ++x) {
1383. solve2[it][y][x+1] = itsMatrix[x][y];
1384. }
1385. }
1386. }
1387. }
1388. }
1389. **else** {
1390. **for** (**var** it = 1 ; it <= 2\*K ; ++it) {
1391. solve2.push(copyM(emptyGrid));
1392. **var** curT = it\*h/2;
1394. *//обрабатываем граничные условия*
1395. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1396. **var** curY = y\*hy;
1397. solve2[it][y][0] = phi\_x0(curY, curT);
1398. solve2[it][y][Nx] = phi\_xl(curY, curT);
1399. }
1401. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1402. **var** curX = x\*hx;
1403. solve2[it][0][x] = phi\_y0(curX, curT);
1404. solve2[it][Ny][x] = phi\_yl(curX, curT);
1405. }
1407. **var** itsMatrix = [];
1409. *//нечетный шаг схемы дробных шагов*
1411. **if** (it%2 == 1) {
1412. **for** (**var** j = 1 ; j < Ny ; ++j) {
1413. **var** curY = j\*hy;
1414. **var** solveMatr = [];
1415. **var** resVector = [];
1416. **var** BVector = [];
1417. **for** (**var** i = 0 ; i <= Nx ; ++i) {
1418. **var** curX = i\*hx;
1419. **var** freeVector = [];
1421. **for** (**var** ii = 0 ; ii <= Nx ; ++ii) {
1422. **if** (i == 0) {
1423. **if** (ii == 0) {
1424. freeVector.push(1);
1425. }
1426. **else** {
1427. freeVector.push(0);
1428. }
1429. }
1430. **else** **if** (i < Nx) {
1431. **if** (ii < i - 1) {
1432. freeVector.push(0);
1433. }
1434. **else** **if** (ii == i - 1) {
1435. **var** A = h / (hx \* hx);
1436. freeVector.push(A);
1437. }
1438. **else** **if** (ii == i) {
1439. **var** B = -1 - (2 \* h / (hx \* hx));
1440. freeVector.push(B);
1441. }
1442. **else** **if** (ii == i + 1) {
1443. **var** C = h / (hx \* hx);
1444. freeVector.push(C);
1445. }
1446. **else** {
1447. freeVector.push(0);
1448. }
1449. }
1450. **else** {
1451. **if** (ii == Nx) {
1452. freeVector.push(1);
1453. }
1454. **else** {
1455. freeVector.push(0);
1456. }
1457. }
1458. }
1459. solveMatr.push(freeVector);
1461. **if** (i == 0) {
1462. BVector.push(phi\_x0(curY, curT-h/2));
1463. }
1464. **else** **if** (i < Nx) {
1465. **var** D = -h\*f(curX, curY, curT-h/2)/2 - solve2[it-1][j][i];
1467. BVector.push(D);
1468. }
1469. **else** {
1470. BVector.push(phi\_xl(curY, curT-h/2));
1471. }
1472. }
1474. resVector = progonka(solveMatr, BVector);
1475. itsMatrix.push(copyV(resVector));
1476. }

1479. **for** (**var** y = 0 ; y < Ny - 1 ; ++y) {
1480. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1481. solve2[it][y+1][x] = itsMatrix[y][x];
1482. }
1483. }
1484. }
1485. **else** {
1486. **for** (**var** j = 1 ; j < Nx ; ++j) {
1487. **var** curX = j\*hx;
1488. **var** solveMatr = [];
1489. **var** resVector = [];
1490. **var** BVector = [];
1491. **for** (**var** i = 0 ; i <= Ny ; ++i) {
1492. **var** curY = i\*hy;
1493. **var** freeVector = [];
1495. **for** (**var** ii = 0 ; ii <= Ny ; ++ii) {
1496. **if** (i == 0) {
1497. **if** (ii == 0) {
1498. freeVector.push(1);
1499. }
1500. **else** {
1501. freeVector.push(0);
1502. }
1503. }
1504. **else** **if** (i < Ny) {
1505. **if** (ii < i - 1) {
1506. freeVector.push(0);
1507. }
1508. **else** **if** (ii == i - 1) {
1509. **var** A = h / (hy \* hy);
1510. freeVector.push(A);
1511. }
1512. **else** **if** (ii == i) {
1513. **var** B = -1 - (2\*h / (hy \* hy));
1514. freeVector.push(B);
1515. }
1516. **else** **if** (ii == i + 1) {
1517. **var** C = h / (hy \* hy);
1518. freeVector.push(C);
1519. }
1520. **else** {
1521. freeVector.push(0);
1522. }
1523. }
1524. **else** {
1525. **if** (ii == Ny) {
1526. freeVector.push(1);
1527. }
1528. **else** {
1529. freeVector.push(0);
1530. }
1531. }
1532. }
1533. solveMatr.push(copyV(freeVector));
1535. **if** (i == 0) {
1536. BVector.push(phi\_y0(curX, curT));
1537. }
1538. **else** **if** (i < Ny) {
1539. **var** D = -h\*f(curX, curY, curT)/2 - solve2[it-1][i][j];
1541. BVector.push(D);
1542. }
1543. **else** {
1544. BVector.push(phi\_yl(curX, curT));
1545. }
1546. }
1548. resVector = progonka(solveMatr, BVector);

1551. itsMatrix.push(copyV(resVector));
1552. }


1556. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1557. **for** (**var** x = 0 ; x < Nx - 1 ; ++x) {
1558. solve2[it][y][x+1] = itsMatrix[x][y];
1559. }
1560. }
1561. }
1562. }
1563. }







1572. **for** (**var** m = 0 ; m <= 2\*K ; m+=2) {
1573. solve.push(copyM(solve2[m]));
1574. }

1577. *//update minMax*
1578. **for** (**var** i = 0 ; i <= K ; ++i) {
1580. **for** (**var** j = 0 ; j <= Ny ; ++j) {
1582. **for** (**var** k = 0 ; k <= Nx ; ++k) {
1584. maxValsyt[i][j] = (solve[i][j][k] > maxValsyt[i][j] ? solve[i][j][k] : maxValsyt[i][j]);
1585. minValsyt[i][j] = (solve[i][j][k] < minValsyt[i][j] ? solve[i][j][k] : minValsyt[i][j]);

1588. maxValsxt[i][k] = (solve[i][j][k] > maxValsxt[i][k] ? solve[i][j][k] : maxValsxt[i][k]);
1589. minValsxt[i][k] = (solve[i][j][k] < minValsxt[i][k] ? solve[i][j][k] : minValsxt[i][k]);
1591. maxValsxy[j][k] = (solve[i][j][k] > maxValsxy[j][k] ? solve[i][j][k] : maxValsxy[j][k]);
1592. minValsxy[j][k] = (solve[i][j][k] < minValsxy[j][k] ? solve[i][j][k] : minValsxy[j][k]);
1594. }
1595. }
1596. }


1600. maxMaxValyt = maxValsyt[0][0];
1601. minMinValyt = minValsyt[0][0];
1603. **for** (**var** t = 0; t <= K ; ++t) {
1604. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1605. maxMaxValyt = (maxValsyt[t][y] > maxMaxValyt ? maxValsyt[t][y] : maxMaxValyt);
1606. minMinValyt = (minValsyt[t][y] < minMinValyt ? minValsyt[t][y] : minMinValyt);
1607. }
1608. }
1610. maxMaxValxt = maxValsxt[0][0];
1611. minMinValxt = minValsxt[0][0];
1613. **for** (**var** t = 0; t <= K ; ++t) {
1614. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1615. maxMaxValxt = (maxValsxt[t][x] > maxMaxValxt ? maxValsxt[t][x] : maxMaxValxt);
1616. minMinValxt = (minValsxt[t][x] < minMinValxt ? minValsxt[t][x] : minMinValxt);
1617. }
1618. }

1621. maxMaxValxy = maxValsxy[0][0];
1622. minMinValxy = minValsxy[0][0];
1624. **for** (**var** y = 0; y <= Ny ; ++y) {
1625. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1626. maxMaxValxy = (maxValsxy[y][x] > maxMaxValxy ? maxValsxy[y][x] : maxMaxValxy);
1627. minMinValxy = (minValsxy[y][x] < minMinValxy ? minValsxy[y][x] : minMinValxy);
1628. }
1629. }

1632. *// error*
1634. maxMaxValE1yt = 1;
1635. minMinValE1yt = 0;
1637. maxMaxValE1xt = 1;
1638. minMinValE1xt = 0;
1640. maxMaxValE1xy = 1;
1641. minMinValE1xy = 0;
1643. maxValsE1yt = [];
1644. minValsE1yt = [];
1646. maxValsE1xt = [];
1647. minValsE1xt = [];
1649. maxValsE1xy = [];
1650. minValsE1xy = [];
1652. EotTx = [];
1653. EotTy = [];
1654. EotTt = [];
1656. EotTx2 = [];
1657. EotTy2 = [];
1658. EotTt2 = [];

1661. errorCube = copyC(solve);
1663. **for** (**var** t = 0 ; t <= K ; ++t) {
1664. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1665. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1666. errorCube[t][y][x] -= etalon[t][y][x];
1667. }
1668. }
1669. }
1671. **for** (**var** t = 0 ; t <= K ; ++t) {
1672. maxValsE1yt.push([]);
1673. minValsE1yt.push([]);
1674. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1675. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1676. **if** (x == 0) {
1677. maxValsE1yt[t].push(errorCube[t][y][x]);
1678. minValsE1yt[t].push(errorCube[t][y][x]);
1679. }
1680. **else** {
1681. maxValsE1yt[t][y] = (errorCube[t][y][x] > maxValsE1yt[t][y] ? errorCube[t][y][x] : maxValsE1yt[t][y]);
1682. minValsE1yt[t][y] = (errorCube[t][y][x] < minValsE1yt[t][y] ? errorCube[t][y][x] : minValsE1yt[t][y]);
1683. }
1684. }
1685. }
1686. }
1688. maxMaxValE1yt = maxValsE1yt[0][0];
1689. minMinValE1yt = minValsE1yt[0][0];
1691. **for** (**var** t = 0; t <= K ; ++t) {
1692. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1693. maxMaxValE1yt = (maxValsE1yt[t][y] > maxMaxValE1yt ? maxValsE1yt[t][y] : maxMaxValE1yt);
1694. minMinValE1yt = (minValsE1yt[t][y] < minMinValE1yt ? minValsE1yt[t][y] : minMinValE1yt);
1695. }
1696. }


1700. **for** (**var** t = 0 ; t <= K ; ++t) {
1701. maxValsE1xt.push([]);
1702. minValsE1xt.push([]);
1703. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1704. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1705. **if** (y == 0) {
1706. maxValsE1xt[t].push(errorCube[t][y][x]);
1707. minValsE1xt[t].push(errorCube[t][y][x]);
1708. }
1709. **else** {
1710. maxValsE1xt[t][x] = (errorCube[t][y][x] > maxValsE1xt[t][x] ? errorCube[t][y][x] : maxValsE1xt[t][x]);
1711. minValsE1xt[t][x] = (errorCube[t][y][x] < minValsE1xt[t][x] ? errorCube[t][y][x] : minValsE1xt[t][x]);
1712. }
1713. }
1714. }
1715. }
1717. maxMaxValE1xt = maxValsE1xt[0][0];
1718. minMinValE1xt = minValsE1xt[0][0];
1720. **for** (**var** t = 0; t <= K ; ++t) {
1721. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1722. maxMaxValE1xt = (maxValsE1xt[t][x] > maxMaxValE1xt ? maxValsE1xt[t][x] : maxMaxValE1xt);
1723. minMinValE1xt = (minValsE1xt[t][x] < minMinValE1xt ? minValsE1xt[t][x] : minMinValE1xt);
1724. }
1725. }

1728. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1729. maxValsE1xy.push([]);
1730. minValsE1xy.push([]);
1731. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1732. **for** (**var** t = 0 ; t <= K ; ++t) {
1733. **if** (t == 0) {
1734. maxValsE1xy[y].push(errorCube[t][y][x]);
1735. minValsE1xy[y].push(errorCube[t][y][x]);
1736. }
1737. **else** {
1738. maxValsE1xy[y][x] = (errorCube[t][y][x] > maxValsE1xy[y][x] ? errorCube[t][y][x] : maxValsE1xy[y][x]);
1739. minValsE1xy[y][x] = (errorCube[t][y][x] < minValsE1xy[y][x] ? errorCube[t][y][x] : minValsE1xy[y][x]);
1740. }
1741. }
1742. }
1743. }
1745. maxMaxValE1xy = maxValsE1xy[0][0];
1746. minMinValE1xy = minValsE1xy[0][0];
1748. **for** (**var** y = 0; y <= Ny ; ++y) {
1749. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1750. maxMaxValE1xy = (maxValsE1xy[y][x] > maxMaxValE1xy ? maxValsE1xy[y][x] : maxMaxValE1xy);
1751. minMinValE1xy = (minValsE1xy[y][x] < minMinValE1xy ? minValsE1xy[y][x] : minMinValE1xy);
1752. }
1753. }


1757. *//eotT*
1759. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1760. **for** (**var** t = 0 ; t <= K ; ++t) {
1761. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1762. **if** (y == 0) {
1763. EotTx.push(errorCube[t][y][x]);
1764. }
1765. **else** {
1766. EotTx[x] = (errorCube[t][y][x] > EotTx[x] ? errorCube[t][y][x] : EotTx[x]);
1767. }
1769. **if** (t == 0) {
1770. EotTx2.push(errorCube[t][y][x]);
1771. }
1772. **else** {
1773. EotTx2[x] = (errorCube[t][y][x] > EotTx2[x] ? errorCube[t][y][x] : EotTx2[x]);
1774. }
1775. }
1776. }
1777. }
1779. minEotTx = EotTx[0];
1780. maxEotTx = EotTx[0];
1782. minEotTx2 = EotTx2[0];
1783. maxEotTx2 = EotTx2[0];
1785. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1786. minEotTx = (EotTx[x] < minEotTx ? EotTx[x] : minEotTx);
1787. minEotTx2 = (EotTx2[x] < minEotTx2 ? EotTx2[x] : minEotTx2);
1789. maxEotTx = (EotTx[x] > maxEotTx ? EotTx[x] : maxEotTx);
1790. maxEotTx2 = (EotTx2[x] > maxEotTx2 ? EotTx2[x] : maxEotTx2);
1791. }

1794. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1795. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1796. **for** (**var** t = 0 ; t <= K ; ++t) {
1797. **if** (t == 0) {
1798. EotTy.push(errorCube[t][y][x]);
1799. }
1800. **else** {
1801. EotTy[y] = (errorCube[t][y][x] > EotTy[y] ? errorCube[t][y][x] : EotTy[y]);
1802. }
1804. **if** (x == 0) {
1805. EotTy2.push(errorCube[t][y][x]);
1806. }
1807. **else** {
1808. EotTy2[y] = (errorCube[t][y][x] > EotTy2[y] ? errorCube[t][y][x] : EotTy2[y]);
1809. }
1810. }
1811. }
1812. }

1815. minEotTy = EotTy[0];
1816. maxEotTy = EotTy[0];
1818. minEotTy2 = EotTy2[0];
1819. maxEotTy2 = EotTy2[0];
1821. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1822. minEotTy = (EotTy[y] < minEotTy ? EotTy[y] : minEotTy);
1823. minEotTy2 = (EotTy2[y] < minEotTy2 ? EotTy2[y] : minEotTy2);
1825. maxEotTy = (EotTy[y] > maxEotTy ? EotTy[y] : maxEotTy);
1826. maxEotTy2 = (EotTy2[y] > maxEotTy2 ? EotTy2[y] : maxEotTy2);
1827. }



1832. **for** (**var** t = 0 ; t <= K ; ++t) {
1833. **for** (**var** x = 0 ; x <= Nx ; ++x) {
1834. **for** (**var** y = 0 ; y <= Ny ; ++y) {
1835. **if** (y == 0) {
1836. EotTt.push(errorCube[t][y][x]);
1837. }
1838. **else** {
1839. EotTt[t] = (errorCube[t][y][x] > EotTt[t] ? errorCube[t][y][x] : EotTt[t]);
1840. }
1842. **if** (x == 0) {
1843. EotTt2.push(errorCube[t][y][x]);
1844. }
1845. **else** {
1846. EotTt2[t] = (errorCube[t][y][x] > EotTt2[t] ? errorCube[t][y][x] : EotTt2[t]);
1847. }
1848. }
1849. }
1850. }
1852. minEotTt = EotTt[0];
1853. maxEotTt = EotTt[0];
1855. minEotTt2 = EotTt2[0];
1856. maxEotTt2 = EotTt2[0];
1858. **for** (**var** t = 0 ; t <= K ; ++t) {
1859. minEotTt = (EotTt[t] < minEotTt ? EotTt[t] : minEotTt);
1860. minEotTt2 = (EotTt2[t] < minEotTt2 ? EotTt2[t] : minEotTt2);
1862. maxEotTt = (EotTt[t] > maxEotTt ? EotTt[t] : maxEotTt);
1863. maxEotTt2 = (EotTt2[t] > maxEotTt2 ? EotTt2[t] : maxEotTt2);
1864. }
1866. makeEtalonGraph();
1867. makeSolveGraph();
1868. makeErrorGraph();
1869. makeErrorTGraph();
1871. go1 = **true**;
1872. go2 = **true**;
1873. }
1875. **function** transfX(x) {
1876. **var** newX;
1877. **if** (setupType == 1) {
1878. newX = 60 + (567 \* (x / lx));
1879. }
1880. **else** **if** (setupType == 2) {
1881. newX = 60 + (567 \* (x / ly));
1882. }
1883. **else** {
1884. newX = 60 + (567 \* (x / T));
1885. }
1886. **return** newX;
1887. }
1889. **function** transfY(y) {
1890. **var** newY = 0;
1891. **if** (setupType == 1) {
1892. **if** (maxValsyt.length > 0 && scale\_type == 2) {
1893. newY = 418.6 - (382.6 \* (y - minValsyt[s3\_cur][s1\_cur])/(maxValsyt[s3\_cur][s1\_cur] - minValsyt[s3\_cur][s1\_cur]));
1894. }
1895. **else** {
1896. newY = 418.6 - (382.6 \* (y - minMinValyt)/(maxMaxValyt - minMinValyt));
1897. }
1898. }
1899. **else** **if** (setupType == 2) {
1900. **if** (maxValsxt.length > 0 && scale\_type == 2) {
1901. newY = 418.6 - (382.6 \* (y - minValsxt[s3\_cur][s2\_cur])/(maxValsxt[s3\_cur][s2\_cur] - minValsxt[s3\_cur][s2\_cur]));
1902. }
1903. **else** {
1904. newY = 418.6 - (382.6 \* (y - minMinValxt)/(maxMaxValxt - minMinValxt));
1905. }
1906. }
1907. **else** **if** (setupType == 3) {
1908. **if** (maxValsxy.length > 0 && scale\_type == 2) {
1909. newY = 418.6 - (382.6 \* (y - minValsxy[s1\_cur][s2\_cur])/(maxValsxy[s1\_cur][s2\_cur] - minValsxy[s1\_cur][s2\_cur]));
1910. }
1911. **else** {
1912. newY = 418.6 - (382.6 \* (y - minMinValxy)/(maxMaxValxy - minMinValxy));
1913. }
1914. }
1915. **return** newY;
1916. }

1919. **function** transfEX(x) {
1920. **var** newX;
1921. **if** (setupType == 1) {
1922. newX = 60.6 + (344.2 \* (x / lx));
1923. }
1924. **else** **if** (setupType == 2) {
1925. newX = 60.6 + (344.2 \* (x / ly));
1926. }
1927. **else** {
1928. newX = 60.6 + (344.2 \* (x / T));
1929. }
1930. **return** newX;
1931. }

1934. **function** transfE2X(x) {
1935. **var** newX;
1936. **if** (pogr\_type <= 2) {
1937. **if** (setupType == 1) {
1938. newX = 60.6 + (344.2 \* (x / T));
1939. }
1940. **else** **if** (setupType == 2) {
1941. newX = 60.6 + (344.2 \* (x / T));
1942. }
1943. **else** {
1944. newX = 60.6 + (344.2 \* (x / lx));
1945. }
1946. }
1947. **else** {
1948. **if** (setupType == 1) {
1949. newX = 60.6 + (344.2 \* (x / ly));
1950. }
1951. **else** **if** (setupType == 2) {
1952. newX = 60.6 + (344.2 \* (x / lx));
1953. }
1954. **else** {
1955. newX = 60.6 + (344.2 \* (x / ly));
1956. }
1957. }
1958. **return** newX;
1959. }
1961. **function** transfEY(y) {
1962. **var** newY = 0;
1963. **if** (setupType == 1) {
1964. **if** (maxValsE.length > 0 && scale\_type == 2) {
1965. newY = 700 - (231 \* (y - minValsE[s1\_cur][s3\_cur])/(maxValsE[s1\_cur][s3\_cur] - minValsE[s1\_cur][s3\_cur]));
1966. }
1967. **else** {
1968. newY = 700 - (231 \* (y - minMinValE)/(maxMaxValE - minMinValE));
1969. }
1970. }
1971. **else** **if** (setupType == 2) {
1972. **if** (maxValsE.length > 0 && scale\_type == 2) {
1973. newY = 700 - (231 \* (y - minValsE[s2\_cur][s3\_cur])/(maxValsE[s2\_cur][s3\_cur] - minValsE[s2\_cur][s3\_cur]));
1974. }
1975. **else** {
1976. newY = 700 - (231 \* (y - minMinValE)/(maxMaxValE - minMinValE));
1977. }
1978. }
1979. **else** **if** (setupType == 3) {
1980. **if** (maxValsE.length > 0 && scale\_type == 2) {
1981. newY = 700 - (231 \* (y - minValsE[s1\_cur][s2\_cur])/(maxValsE[s1\_cur][s2\_cur] - minValsE[s1\_cur][s2\_cur]));
1982. }
1983. **else** {
1984. newY = 700 - (231 \* (y - minMinValE)/(maxMaxValE - minMinValE));
1985. }
1986. }
1987. **return** newY;
1988. }
1990. **function** transfE2Y(y) {
1991. **var** newY = 0;
1992. **if** (setupType == 1) {
1993. newY = 700 - (231 \* (y - minEotTx)/(maxEotTx - minEotTx));
1994. }
1995. **else** **if** (setupType == 2) {
1996. newY = 700 - (231 \* (y - minEotTy)/(maxEotTy - minEotTy));
1997. }
1998. **else** **if** (setupType == 3) {
1999. newY = 700 - (231 \* (y - minEotTt)/(maxEotTt - minEotTt));
2000. }
2002. **return** newY;
2003. }

2006. **function** makeEtalonGraph() {
2007. **for** (**var** i = 0 ; i < Nx ; ++i) {
2008. **var** join = **new** lib.line();
2009. stage.addChild(join);
2010. join.x = transfX(xVector[i]);
2011. join.y = transfY(etalon[0][0][i]);
2012. join.endX = transfX(xVector[i+1]);
2013. join.endY = transfY(etalon[0][0][i+1]);
2015. join.gotoAndStop(0);
2016. join.num = i;
2018. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
2020. join.scaleX = join.len;
2021. join.scaleY = 1;
2023. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
2025. join.gen = generation;

2028. join.visible = **true**;
2029. join.alpha = 1;
2031. join.addEventListener('tick', setPoses11);
2032. }
2033. **for** (**var** i = 0 ; i < Ny ; ++i) {
2034. **var** join = **new** lib.line();
2035. stage.addChild(join);
2036. join.x = transfX(yVector[i]);
2037. join.y = transfY(etalon[0][i][0]);
2038. join.endX = transfX(yVector[i+1]);
2039. join.endY = transfY(etalon[0][i+1][0]);
2041. join.gotoAndStop(0);
2042. join.num = i;
2044. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
2046. join.scaleX = join.len;
2047. join.scaleY = 1;
2049. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
2051. join.gen = generation;

2054. join.visible = **true**;
2055. join.alpha = 1;
2057. join.addEventListener('tick', setPoses12);
2058. }
2059. **for** (**var** i = 0 ; i < K ; ++i) {
2060. **var** join = **new** lib.line();
2061. stage.addChild(join);
2062. join.x = transfX(tVector[i]);
2063. join.y = transfY(etalon[i][0][0]);
2064. join.endX = transfX(tVector[i+1]);
2065. join.endY = transfY(etalon[i+1][0][0]);
2067. join.gotoAndStop(0);
2068. join.num = i;
2070. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
2072. join.scaleX = join.len;
2073. join.scaleY = 1;
2075. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
2077. join.gen = generation;

2080. join.visible = **true**;
2081. join.alpha = 1;
2083. join.addEventListener('tick', setPoses13);
2084. }
2086. }
2088. **function** makeSolveGraph() {
2089. **for** (**var** i = 0 ; i < Nx ; ++i) {
2090. **var** join = **new** lib.line();
2091. stage.addChild(join);
2092. join.x = transfX(xVector[i]);
2093. join.y = transfY(solve[0][0][i]);
2094. join.endX = transfX(xVector[i+1]);
2095. join.endY = transfY(solve[0][0][i+1]);
2097. join.gotoAndStop(1);
2098. join.num = i;
2100. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
2102. join.scaleX = join.len;
2103. join.scaleY = 1;
2105. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
2107. join.gen = generation;

2110. join.visible = **true**;
2111. join.alpha = 1;
2113. join.addEventListener('tick', setPoses21);
2114. }
2115. **for** (**var** i = 0 ; i < Ny ; ++i) {
2116. **var** join = **new** lib.line();
2117. stage.addChild(join);
2118. join.x = transfX(yVector[i]);
2119. join.y = transfY(solve[0][i][0]);
2120. join.endX = transfX(yVector[i+1]);
2121. join.endY = transfY(solve[0][i+1][0]);
2123. join.gotoAndStop(1);
2124. join.num = i;
2126. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
2128. join.scaleX = join.len;
2129. join.scaleY = 1;
2131. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
2133. join.gen = generation;

2136. join.visible = **true**;
2137. join.alpha = 1;
2139. join.addEventListener('tick', setPoses22);
2140. }
2141. **for** (**var** i = 0 ; i < K ; ++i) {
2142. **var** join = **new** lib.line();
2143. stage.addChild(join);
2144. join.x = transfX(tVector[i]);
2145. join.y = transfY(solve[i][0][0]);
2146. join.endX = transfX(tVector[i+1]);
2147. join.endY = transfY(solve[i+1][0][0]);
2149. join.gotoAndStop(1);
2150. join.num = i;
2152. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
2154. join.scaleX = join.len;
2155. join.scaleY = 1;
2157. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
2159. join.gen = generation;

2162. join.visible = **true**;
2163. join.alpha = 1;
2165. join.addEventListener('tick', setPoses23);
2166. }
2167. }
2169. **function** makeErrorGraph() {
2170. **for** (**var** i = 0 ; i < Nx ; ++i) {
2171. **var** join = **new** lib.line();
2172. stage.addChild(join);
2173. join.x = transfEX(xVector[i]);
2174. join.y = transfEY(errorCube[0][0][i]);
2175. join.endX = transfEX(xVector[i+1]);
2176. join.endY = transfEY(errorCube[0][0][i+1]);
2178. join.gotoAndStop(3);
2179. join.num = i;
2181. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
2183. join.scaleX = join.len;
2184. join.scaleY = 1;
2186. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
2188. join.gen = generation;

2191. join.visible = **true**;
2192. join.alpha = 1;
2194. join.addEventListener('tick', setPoses31);
2195. }
2196. **for** (**var** i = 0 ; i < Ny ; ++i) {
2197. **var** join = **new** lib.line();
2198. stage.addChild(join);
2199. join.x = transfEX(yVector[i]);
2200. join.y = transfEY(errorCube[0][i][0]);
2201. join.endX = transfEX(yVector[i+1]);
2202. join.endY = transfEY(errorCube[0][i+1][0]);
2204. join.gotoAndStop(3);
2205. join.num = i;
2207. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
2209. join.scaleX = join.len;
2210. join.scaleY = 1;
2212. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
2214. join.gen = generation;

2217. join.visible = **true**;
2218. join.alpha = 1;
2220. join.addEventListener('tick', setPoses32);
2221. }
2222. **for** (**var** i = 0 ; i < K ; ++i) {
2223. **var** join = **new** lib.line();
2224. stage.addChild(join);
2225. join.x = transfEX(tVector[i]);
2226. join.y = transfEY(errorCube[i][0][0]);
2227. join.endX = transfEX(tVector[i+1]);
2228. join.endY = transfEY(errorCube[i+1][0][0]);
2230. join.gotoAndStop(3);
2231. join.num = i;
2233. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
2235. join.scaleX = join.len;
2236. join.scaleY = 1;
2238. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
2240. join.gen = generation;

2243. join.visible = **true**;
2244. join.alpha = 1;
2246. join.addEventListener('tick', setPoses33);
2247. }

2250. }

2253. **function** makeErrorTGraph() {
2254. **for** (**var** i = 0 ; i < Nx ; ++i) {
2255. **var** join = **new** lib.line();
2256. stage.addChild(join);
2257. join.x = transfE2X(xVector[i]);
2258. join.y = transfE2Y(EotTx[i]);
2259. join.endX = transfE2X(xVector[i+1]);
2260. join.endY = transfE2Y(EotTx[i+1]);
2262. join.gotoAndStop(3);
2263. join.num = i;
2265. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
2267. join.scaleX = join.len;
2268. join.scaleY = 1;
2270. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
2272. join.gen = generation;

2275. join.visible = **true**;
2276. join.alpha = 1;
2278. join.addEventListener('tick', setPoses41);
2279. }
2280. **for** (**var** i = 0 ; i < Ny ; ++i) {
2281. **var** join = **new** lib.line();
2282. stage.addChild(join);
2283. join.x = transfE2X(yVector[i]);
2284. join.y = transfE2Y(EotTy[i]);
2285. join.endX = transfE2X(yVector[i+1]);
2286. join.endY = transfE2Y(EotTy[i+1]);
2288. join.gotoAndStop(3);
2289. join.num = i;
2291. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
2293. join.scaleX = join.len;
2294. join.scaleY = 1;
2296. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
2298. join.gen = generation;

2301. join.visible = **true**;
2302. join.alpha = 1;
2304. join.addEventListener('tick', setPoses42);
2305. }
2306. **for** (**var** i = 0 ; i < K ; ++i) {
2307. **var** join = **new** lib.line();
2308. stage.addChild(join);
2309. join.x = transfE2X(tVector[i]);
2310. join.y = transfE2Y(EotTt[i]);
2311. join.endX = transfE2X(tVector[i+1]);
2312. join.endY = transfE2Y(EotTt[i+1]);
2314. join.gotoAndStop(3);
2315. join.num = i;
2317. join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX - join.x), 2));
2319. join.scaleX = join.len;
2320. join.scaleY = 1;
2322. join.rotation = Math.atan2((join.endY - join.y), (join.endX - join.x)) \* 180 / Math.PI;
2324. join.gen = generation;

2327. join.visible = **true**;
2328. join.alpha = 1;
2330. join.addEventListener('tick', setPoses43);
2331. }
2332. }


2336. **function** setPoses11(e) {
2337. **var** object = e.currentTarget;
2338. **if** (object.gen == generation) {
2339. object.x = transfX(xVector[object.num]);
2340. object.y = transfY(etalon[s3\_cur][s1\_cur][object.num]);
2341. object.endX = transfX(xVector[object.num+1]);
2342. object.endY = transfY(etalon[s3\_cur][s1\_cur][object.num+1]);
2344. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
2346. **if** (setupType == 1) {
2347. object.visible = **true**;
2348. }
2349. **else** {
2350. object.visible = **false**;
2351. }

2354. object.scaleX = object.len;
2355. object.scaleY = 1;
2357. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
2358. }
2359. **else** **if** (object.gen != generation) {
2360. object.alpha -= 3/30;
2361. }
2363. **if** (object.alpha <= 0) {
2364. object.alpha = 0;
2365. object.visible = **false**;
2366. object.removeEventListener('tick', setPoses11);
2367. stage.removeChild(object);
2368. }
2370. }
2372. **function** setPoses12(e) {
2373. **var** object = e.currentTarget;
2374. **if** (object.gen == generation) {
2375. object.x = transfX(yVector[object.num]);
2376. object.y = transfY(etalon[s3\_cur][object.num][s2\_cur]);
2377. object.endX = transfX(yVector[object.num+1]);
2378. object.endY = transfY(etalon[s3\_cur][object.num+1][s2\_cur]);
2380. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
2382. **if** (setupType == 2) {
2383. object.visible = **true**;
2384. }
2385. **else** {
2386. object.visible = **false**;
2387. }

2390. object.scaleX = object.len;
2391. object.scaleY = 1;
2393. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
2394. }
2395. **else** **if** (object.gen != generation) {
2396. object.alpha -= 3/30;
2397. }
2399. **if** (object.alpha <= 0) {
2400. object.alpha = 0;
2401. object.visible = **false**;
2402. object.removeEventListener('tick', setPoses12);
2403. stage.removeChild(object);
2404. }
2406. }
2408. **function** setPoses13(e) {
2409. **var** object = e.currentTarget;
2410. **if** (object.gen == generation) {
2411. object.x = transfX(tVector[object.num]);
2412. object.y = transfY(etalon[object.num][s1\_cur][s2\_cur]);
2413. object.endX = transfX(tVector[object.num+1]);
2414. object.endY = transfY(etalon[object.num+1][s1\_cur][s2\_cur]);
2416. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
2418. **if** (setupType == 3) {
2419. object.visible = **true**;
2420. }
2421. **else** {
2422. object.visible = **false**;
2423. }

2426. object.scaleX = object.len;
2427. object.scaleY = 1;
2429. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
2430. }
2431. **else** **if** (object.gen != generation) {
2432. object.alpha -= 3/30;
2433. }
2435. **if** (object.alpha <= 0) {
2436. object.alpha = 0;
2437. object.visible = **false**;
2438. object.removeEventListener('tick', setPoses13);
2439. stage.removeChild(object);
2440. }
2442. }


2446. **function** setPoses21(e) {
2447. **var** object = e.currentTarget;
2448. **if** (object.gen == generation) {
2449. object.x = transfX(xVector[object.num]);
2450. object.y = transfY(solve[s3\_cur][s1\_cur][object.num]);
2451. object.endX = transfX(xVector[object.num+1]);
2452. object.endY = transfY(solve[s3\_cur][s1\_cur][object.num+1]);
2454. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
2456. **if** (setupType == 1) {
2457. object.visible = **true**;
2458. }
2459. **else** {
2460. object.visible = **false**;
2461. }

2464. object.scaleX = object.len;
2465. object.scaleY = 1;
2467. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
2468. }
2469. **else** **if** (object.gen != generation) {
2470. object.alpha -= 3/30;
2471. }
2473. **if** (object.alpha <= 0) {
2474. object.alpha = 0;
2475. object.visible = **false**;
2476. object.removeEventListener('tick', setPoses21);
2477. stage.removeChild(object);
2478. }
2480. }
2482. **function** setPoses22(e) {
2483. **var** object = e.currentTarget;
2484. **if** (object.gen == generation) {
2485. object.x = transfX(yVector[object.num]);
2486. object.y = transfY(solve[s3\_cur][object.num][s2\_cur]);
2487. object.endX = transfX(yVector[object.num+1]);
2488. object.endY = transfY(solve[s3\_cur][object.num+1][s2\_cur]);
2490. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
2492. **if** (setupType == 2) {
2493. object.visible = **true**;
2494. }
2495. **else** {
2496. object.visible = **false**;
2497. }

2500. object.scaleX = object.len;
2501. object.scaleY = 1;
2503. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
2504. }
2505. **else** **if** (object.gen != generation) {
2506. object.alpha -= 3/30;
2507. }
2509. **if** (object.alpha <= 0) {
2510. object.alpha = 0;
2511. object.visible = **false**;
2512. object.removeEventListener('tick', setPoses22);
2513. stage.removeChild(object);
2514. }
2516. }
2518. **function** setPoses23(e) {
2519. **var** object = e.currentTarget;
2520. **if** (object.gen == generation) {
2521. object.x = transfX(tVector[object.num]);
2522. object.y = transfY(solve[object.num][s1\_cur][s2\_cur]);
2523. object.endX = transfX(tVector[object.num+1]);
2524. object.endY = transfY(solve[object.num+1][s1\_cur][s2\_cur]);
2526. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
2528. **if** (setupType == 3) {
2529. object.visible = **true**;
2530. }
2531. **else** {
2532. object.visible = **false**;
2533. }

2536. object.scaleX = object.len;
2537. object.scaleY = 1;
2539. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
2540. }
2541. **else** **if** (object.gen != generation) {
2542. object.alpha -= 3/30;
2543. }
2545. **if** (object.alpha <= 0) {
2546. object.alpha = 0;
2547. object.visible = **false**;
2548. object.removeEventListener('tick', setPoses23);
2549. stage.removeChild(object);
2550. }
2552. }

2555. **function** setPoses31(e) {
2556. **var** object = e.currentTarget;
2557. **if** (object.gen == generation) {
2558. object.x = transfEX(xVector[object.num]);
2559. object.y = transfEY(errorCube[s3\_cur][s1\_cur][object.num]);
2560. object.endX = transfEX(xVector[object.num+1]);
2561. object.endY = transfEY(errorCube[s3\_cur][s1\_cur][object.num+1]);
2563. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
2565. **if** (setupType == 1 && pogr\_type == 1) {
2566. object.visible = **true**;
2567. }
2568. **else** {
2569. object.visible = **false**;
2570. }

2573. object.scaleX = object.len;
2574. object.scaleY = 1;
2576. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
2577. }
2578. **else** **if** (object.gen != generation) {
2579. object.alpha -= 3/30;
2580. }
2582. **if** (object.alpha <= 0) {
2583. object.alpha = 0;
2584. object.visible = **false**;
2585. object.removeEventListener('tick', setPoses21);
2586. stage.removeChild(object);
2587. }
2589. }
2591. **function** setPoses32(e) {
2592. **var** object = e.currentTarget;
2593. **if** (object.gen == generation) {
2594. object.x = transfEX(yVector[object.num]);
2595. object.y = transfEY(errorCube[s3\_cur][object.num][s2\_cur]);
2596. object.endX = transfEX(yVector[object.num+1]);
2597. object.endY = transfEY(errorCube[s3\_cur][object.num+1][s2\_cur]);
2599. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
2601. **if** (setupType == 2 && pogr\_type == 1) {
2602. object.visible = **true**;
2603. }
2604. **else** {
2605. object.visible = **false**;
2606. }

2609. object.scaleX = object.len;
2610. object.scaleY = 1;
2612. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
2613. }
2614. **else** **if** (object.gen != generation) {
2615. object.alpha -= 3/30;
2616. }
2618. **if** (object.alpha <= 0) {
2619. object.alpha = 0;
2620. object.visible = **false**;
2621. object.removeEventListener('tick', setPoses22);
2622. stage.removeChild(object);
2623. }
2625. }
2627. **function** setPoses33(e) {
2628. **var** object = e.currentTarget;
2629. **if** (object.gen == generation) {
2630. object.x = transfEX(tVector[object.num]);
2631. object.y = transfEY(errorCube[object.num][s1\_cur][s2\_cur]);
2632. object.endX = transfEX(tVector[object.num+1]);
2633. object.endY = transfEY(errorCube[object.num+1][s1\_cur][s2\_cur]);
2635. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
2637. **if** (setupType == 3 && pogr\_type == 1) {
2638. object.visible = **true**;
2639. }
2640. **else** {
2641. object.visible = **false**;
2642. }

2645. object.scaleX = object.len;
2646. object.scaleY = 1;
2648. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
2649. }
2650. **else** **if** (object.gen != generation) {
2651. object.alpha -= 3/30;
2652. }
2654. **if** (object.alpha <= 0) {
2655. object.alpha = 0;
2656. object.visible = **false**;
2657. object.removeEventListener('tick', setPoses23);
2658. stage.removeChild(object);
2659. }
2661. }

2664. **function** setPoses41(e) {
2665. **var** object = e.currentTarget;
2666. **if** (object.gen == generation) {
2667. object.x = transfE2X(xVector[object.num]);
2668. object.endX = transfE2X(xVector[object.num+1]);
2670. **if** (setupType == 2 && pogr\_type == 3) {
2671. object.y = transfEY(EotTx[object.num]);
2672. object.endY = transfEY(EotTx[object.num+1]);
2673. }
2674. **else** {
2675. object.y = transfEY(EotTx2[object.num]);
2676. object.endY = transfEY(EotTx2[object.num+1]);
2677. }
2679. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
2681. **if** ((setupType == 2 && pogr\_type == 3) || (setupType == 3 && pogr\_type == 2)) {
2682. object.visible = **true**;
2683. }
2684. **else** {
2685. object.visible = **false**;
2686. }

2689. object.scaleX = object.len;
2690. object.scaleY = 1;
2692. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
2693. }
2694. **else** **if** (object.gen != generation) {
2695. object.alpha -= 3/30;
2696. }
2698. **if** (object.alpha <= 0) {
2699. object.alpha = 0;
2700. object.visible = **false**;
2701. object.removeEventListener('tick', setPoses41);
2702. stage.removeChild(object);
2703. }
2705. }

2708. **function** setPoses42(e) {
2709. **var** object = e.currentTarget;
2710. **if** (object.gen == generation) {
2711. object.x = transfE2X(yVector[object.num]);
2712. object.endX = transfE2X(yVector[object.num+1]);
2714. **if** (setupType == 1 && pogr\_type == 3) {
2715. object.y = transfEY(EotTy[object.num]);
2716. object.endY = transfEY(EotTy[object.num+1]);
2717. }
2718. **else** {
2719. object.y = transfEY(EotTy2[object.num]);
2720. object.endY = transfEY(EotTy2[object.num+1]);
2721. }
2723. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
2725. **if** ((setupType == 1 && pogr\_type == 3) || (setupType == 3 && pogr\_type == 3)) {
2726. object.visible = **true**;
2727. }
2728. **else** {
2729. object.visible = **false**;
2730. }

2733. object.scaleX = object.len;
2734. object.scaleY = 1;
2736. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
2737. }
2738. **else** **if** (object.gen != generation) {
2739. object.alpha -= 3/30;
2740. }
2742. **if** (object.alpha <= 0) {
2743. object.alpha = 0;
2744. object.visible = **false**;
2745. object.removeEventListener('tick', setPoses42);
2746. stage.removeChild(object);
2747. }
2749. }

2752. **function** setPoses43(e) {
2753. **var** object = e.currentTarget;
2754. **if** (object.gen == generation) {
2755. object.x = transfE2X(tVector[object.num]);
2756. object.endX = transfE2X(tVector[object.num+1]);
2758. **if** (setupType == 1 && pogr\_type == 2) {
2759. object.y = transfEY(EotTt[object.num]);
2760. object.endY = transfEY(EotTt[object.num+1]);
2761. }
2762. **else** {
2763. object.y = transfEY(EotTt2[object.num]);
2764. object.endY = transfEY(EotTt2[object.num+1]);
2765. }
2767. object.len = Math.sqrt(Math.pow((object.endY - object.y), 2) + Math.pow((object.endX - object.x), 2));
2769. **if** ((setupType == 1 && pogr\_type == 2) || (setupType == 2 && pogr\_type == 2)) {
2770. object.visible = **true**;
2771. }
2772. **else** {
2773. object.visible = **false**;
2774. }

2777. object.scaleX = object.len;
2778. object.scaleY = 1;
2780. object.rotation = Math.atan2((object.endY - object.y), (object.endX - object.x)) \* 180 / Math.PI;
2781. }
2782. **else** **if** (object.gen != generation) {
2783. object.alpha -= 3/30;
2784. }
2786. **if** (object.alpha <= 0) {
2787. object.alpha = 0;
2788. object.visible = **false**;
2789. object.removeEventListener('tick', setPoses43);
2790. stage.removeChild(object);
2791. }
2793. }



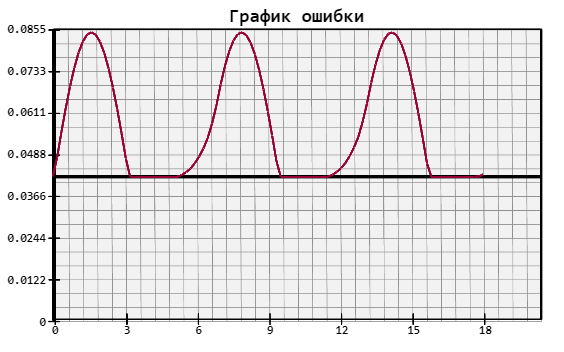


2800. *//метод прогонки*
2801. **function** progonka(matrix, vectorB) {
2802. **var** vectorX = [];
2804. **var** N = vectorB.length;
2806. **var** alphas = [];
2807. **var** betas = [];

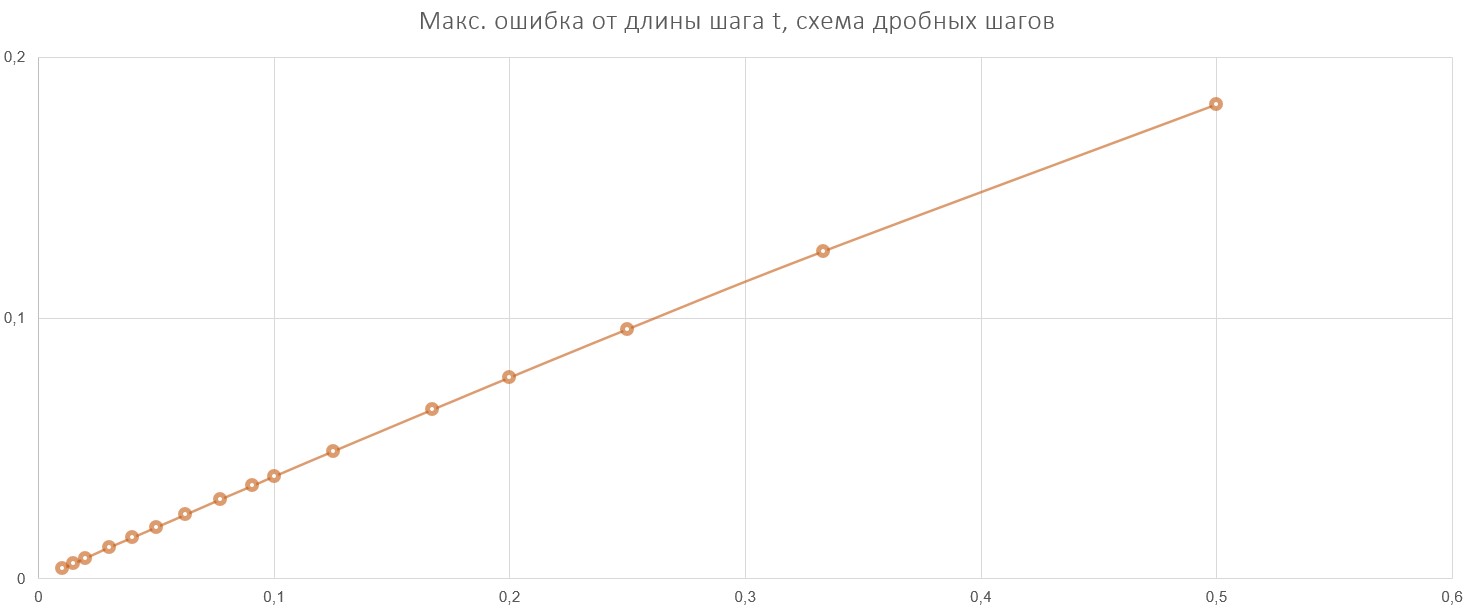
2810. **for** (**var** i = 0 ; i < vectorB.length ; ++i) {
2811. alphas.push(0);
2812. betas.push(0);
2813. }
2815. *//Прямой ход прогонки*
2817. **for** (**var** i = 0; i < N; ++i) {
2818. **var** A0, C0, B0, F0;
2820. **if** (i - 1 < 0) {
2821. A0 = 0;
2822. }
2823. **else** {
2824. A0 = matrix[i][i - 1];
2825. }
2827. C0 = -1 \* matrix[i][i];
2829. **if** (i + 1 < N) {
2830. B0 = matrix[i][i + 1];
2831. }
2832. **else** {
2833. B0 = 0;
2834. }
2836. F0 = vectorB[i];

2839. **if** (i == 0) {
2840. alphas[i] = B0 / C0;
2841. betas[i] = -(F0 / C0);
2842. }
2843. **else** **if** (i == N - 1) {
2844. alphas[i] = 0;
2845. betas[i] = (betas[i - 1] \* A0 - F0) / (C0 - alphas[i - 1] \* A0);
2846. }
2847. **else** {
2848. alphas[i] = B0 / (C0 - alphas[i - 1] \* A0);
2849. betas[i] = (betas[i - 1] \* A0 - F0) / (C0 - alphas[i - 1] \* A0);
2850. }
2852. }

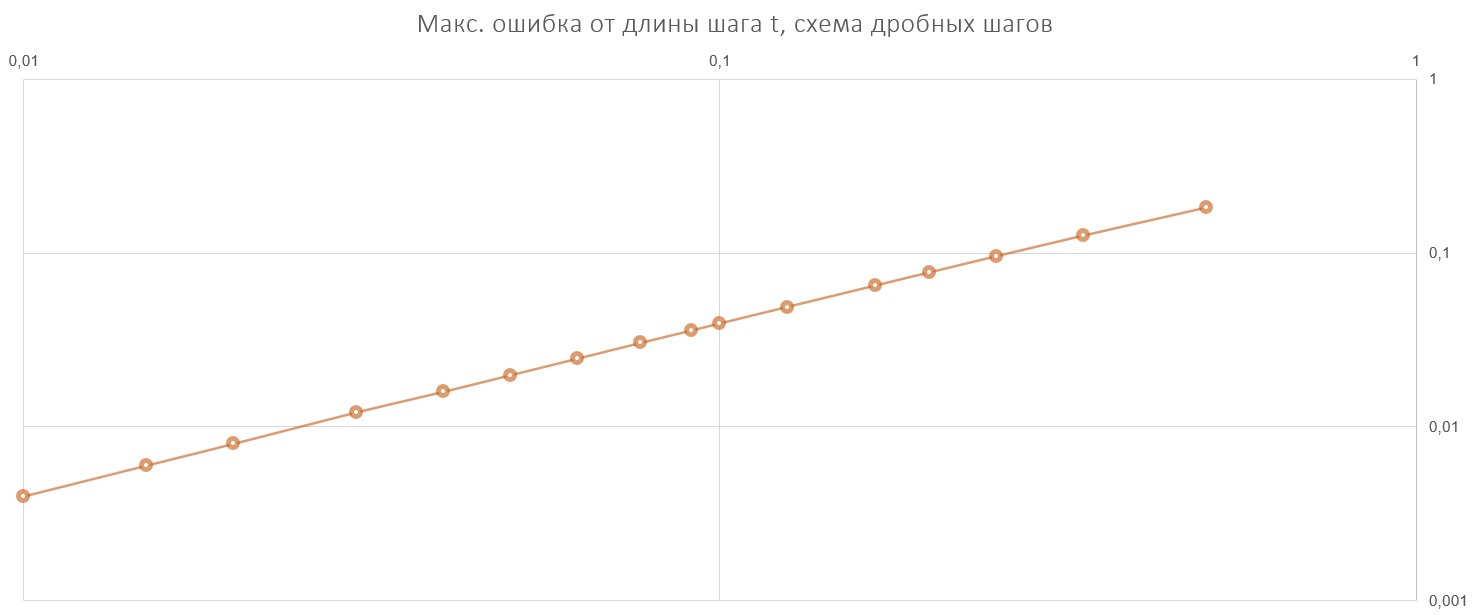
2855. vectorX[N - 1] = betas[N - 1];
2857. **for** (**var** i = 2; i <= N; ++i) {
2858. vectorX[N - i] = alphas[N - i] \* vectorX[N - i + 1] + betas[N - i];
2859. }
2861. **return** vectorX;
2862. }
2863. Графики

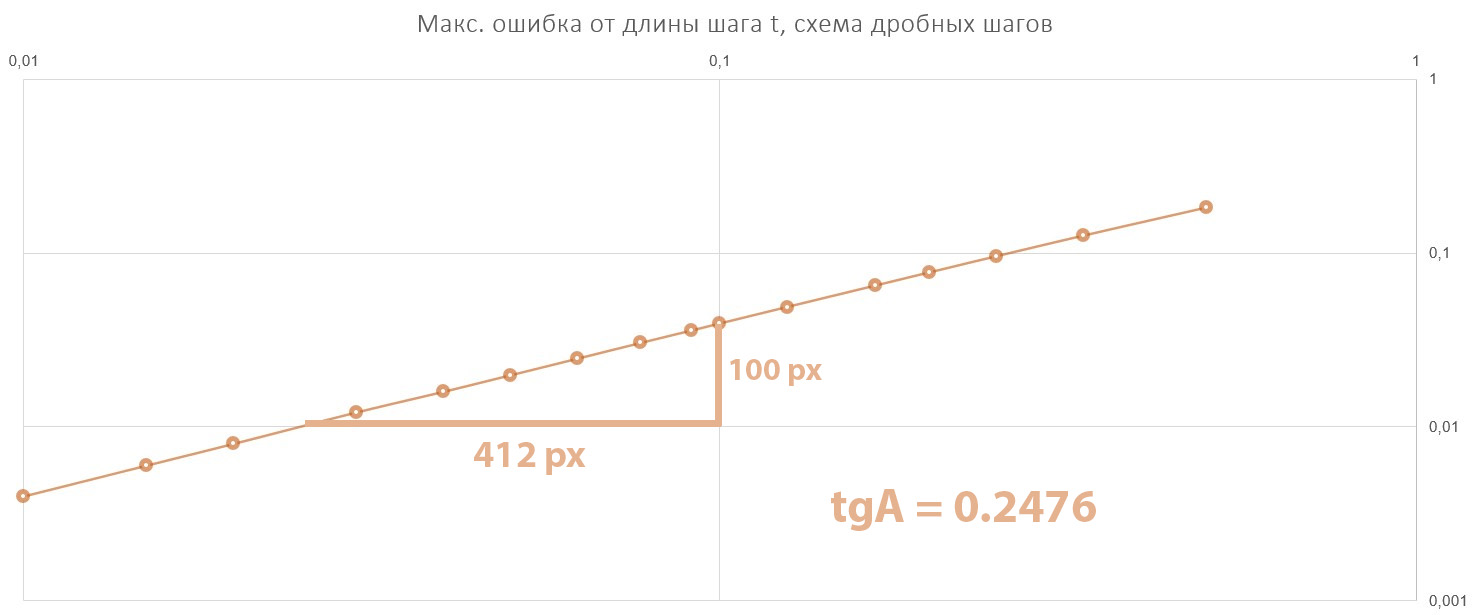


При T = 18



Видим, что ошибка растет линейно с размером шага.





Данная лабораторная работа выполнена: **22 января 2021.**

**Вывод по лабораторным**

Выполняя данные лабораторные, я познакомился с различными конечно-разностными схемами. Они позволяют решить многие задачи математической физики, которые решать аналитически или напрямую будет очень сложно. Однако, результаты не всегда бывают точными, существует хоть какая-то, но погрешность. В зависимости от того, как растет погрешность с шагом, можно определить какой порядок аппроксимации уравнения. На самом деле, было очень сложно больше сделать весь рабочий интерфейс, запрограммировать отрисовку, выбор параметров, динамические графики и графики погрешностей. Все это не сильно связано с самим предметом, однако, это улучшило мои основные навыки программирования и дизайна, что для меня очень важно. Не знаю, придется ли мне еще столкнуться с похожими задачами, но было интересно решать их с помощью схем. Самой лютой и сложной лабораторной оказалась последняя. Примерно 12 часов непрерывной работы (пишу этот отчет в 8 утра и до сих пор даже не спал) и почти 3 тысячи строк кода ушло на то, чтобы запрограммировать ее, сделать рабочий интерфейс, графики, однако, даже сейчас присутствуют небольшие косяки. Пришлось столкнуться с большой проблемой, когда схема в последней лабораторной не работала как нужно. Дело в том, что я с помощью нее только решал середину (методом прогонки), а нужно было весь участок, включая границы. Можно много еще чего написать в выводе, однако, думаю, что эта работа была довольно полезной для меня, хотя бы я усвоил свои навыки и узнал что-то новое.