Санкт-Петербургский государственный университет телекоммуникаций

им. проф. М. А. Бонч-Бруевича

Факультет ИКПИ

**Курсовая работа**

По дисциплине «Программирование»

Выполнил: Кузьмин Артемий

Группа: ИКПИ-85

Вариант работы: 37-7-7

Санкт-Петербург

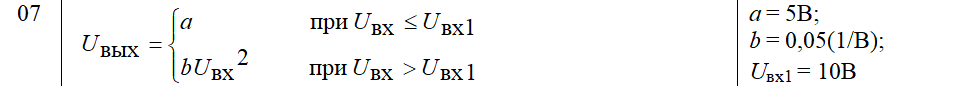
2020

1. Постановка задачи

Необходимо написать программу, позволяющую для данного варианта электрической цепи рассчитать значения входного сигнала, значения выходного сигнала, рассчитать с заданной точностью значение характеристики выходного сигнала, в данном варианте – момент времени, при котором выходной сигнал опускается уровня 0,1\*Umax.

Вариант заданий:





1. Разработка алгоритма

**Алгоритм главного модуля:**

1) при первом проходе, т.е. при N = Nнач, величина погрешности не вычисляется, а только запоминается вычисленное значение характеристики.

2) вводится проверка превышения текущего числа интервалов N некоторого максимального значения Nmax, допустим, 1000. Введение такой проверки предотвращает бесконечно долгое выполнение программы в случае слишком медленного уменьшения относительной погрешности δW. Если количество интервалов превысит максимально допустимое Nmax, то процесс вычисления прекращается и должно быть выдано сообщение “Требуемая точность не достигнута”.

3) предусматривается режим вывода на экран ряда контрольных значений времени и напряжений.

4) выполнена детализация этапа вычислений величины W. Вычисление W реализуется с помощью обращения к трем подпрограммам: формирование массива t и Uвх, формирование массива Uвых формирование величины W.

**Алгоритм подпрограммы, рассчитывающей значения входного напряжения:**

Для записи значений входного напряжения был использован массив типа float размером N+1. Для вычисления этого массива, была использована циклическая конструкция for.

Для выбора формулы вычисления в зависимости от момента времени была использована вложенная условная конструкция if-else. Значение входного сигнала задается графиком, необходимые формулы выведены на листе с ручными расчетами.

**Алгоритм подпрограммы, рассчитывающей значения выходного напряжения:**

Для записи значений входного напряжения был использован массив типа float размером N+1. Для вычисления этого массива, была использована циклическая конструкция for.

Для выбора формулы вычисления в зависимости от момента времени была использована вложенная условная конструкция if-else.

**Алгоритм подпрограммы, рассчитывающей характеристику:**

Для подсчёта используется циклическая конструкция while. В момент, когда значение выходного напряжения меньше, чем C\*Umax, цикл останавливается, считается непосредственно время и выводится. Так как данная момент должен быть в тот время, когда напряжение падает, выводиться дополнительное условие для цикла: в тот момент когда Uвых < С\*umax, дополнительно необходимо проверить, что предшествующее значение Uвых больше или равно С\*umax.

**Алгоритм подпрограммы контрольного вывода:**

В цикле выводятся моменты времени, входное напряжение и выходное напряжение, а потом вывод характеристики и количества интервалов. Эта подпрограмма совершает работу, если значение переменной DEBUG равно “y” или “Y”.

1. Таблица идентификаторов

|  |  |  |  |
| --- | --- | --- | --- |
| Обозначение в задаче | Тип | Обозначение в программе | Назначение |
| t1, t2 | float | t1, t2 | Моменты времени между tнач и tкон, относительно которых проходит вычисление входного напряжения |
| tнач, tкон | t0, tk | Начальный и конечный моменты времени наблюдения выходного напряжения |
| a, b | a, b | Параметры, относительно которых вычисляется входное напряжение |
|  |  | Границы изменения напряжения на графике |
| U­1, U2 | U1, U2 |
| C | C | Параметр |
| dW | dTUMAXC | Погрешность рассчёта характеристики |
| W | tmax | Характеристика в данный проход |
| tmax1 | Характеристика в прошлый проход |
| Uвх, Uвых | Uinp[Nmax+1], Uout[Nmax+1] | Массивы входного и выходного напряжения |
| E | eps | Эпсилон, допустимая погрешность вычисления параметра |
| N, N­нач | int | N, N0 | Текущее и начальное значение интервалов времени |
|  | char | DEBUG | Переменная, проверяющая, входить ли в режим контрольного вывода |

1. Проверка работоспособности

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Набор значений | |  | Ручной рассчёт | | | | Ручной рассчёт | | |
| Перем. | Знач. | T | Uвх | Uвых | Uвых < C\*Umax | Uвх | | Uвых | t, когда Uвых<C\*Umax |
| a | 5 | 10 | 0 | 5 |  | 0 | | 5 | 87 |
| b | 0,05 | 20 | 8 | 5 |  | 8 | | 20 |
| c | 0.1 | 22 | 100 | 500 |  | 100 | | 20 |
| tнач | 10 | 30 | 96.7 | 467.2 |  | 96.666 | | 467.222 |
| t1 | 22 | 40 | 92.5 | 428 |  | 92.5 | | 427.877 |
| t2 | 70 | 50 | 88.3 | 390 |  | 88.333 | | 390.13 | 87 |
| tкон | 90 | 60 | 84 | 354.2 |  | 84.166 | | 354.201 |
| U1 | 100 | 70 | 80 | 320 |  | 80 | | 320 |
| U2 | 80 | 80 | 40 | 80 |  | 40 | | 80 |
| eps | 0,01 | 86 | 16 | 12 |  | 16 | | 12 |
|  | | 87 | 12 | 7 | + | 12 | | 7 |
| 88 | 8 | 5 |  | 8 | | 8 |
| 89 | 4 | 5 |  | 4 | | 5 |
| 90 | 0 | 0 |  | 0 | | 0 |
|  | | | | | | |
|  | | |

Для проверки режима аномального завершения программы была задана допустимая погрешность eps = −1. Поскольку вычисляемая в программе относительная погрешность характеристики всегда положительна, то она будет больше допустимой при любых значениях *N*max.  


1. Текст программы:

**Файл main.c:**

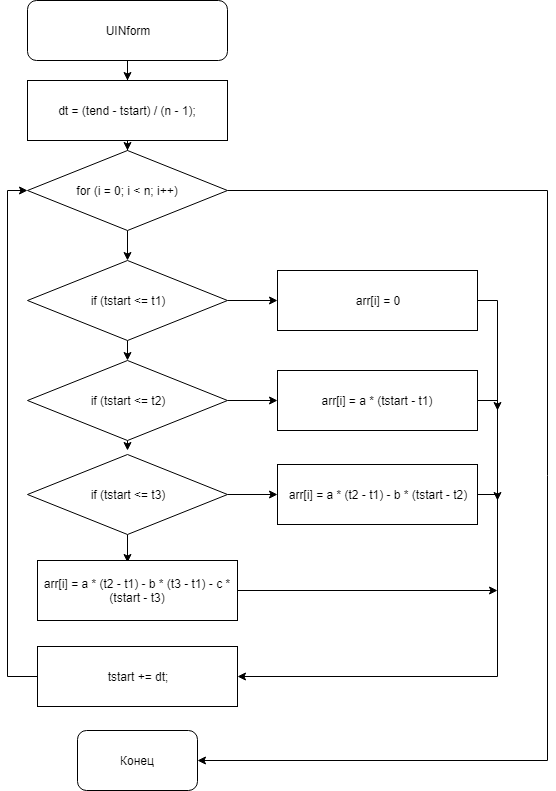
1. #define \_CRT\_SECURE\_NO\_WARNINGS
2. #define Nmax 1000
3. #include <stdio.h>
4. #include <math.h>
5. #include "func.h"
6. #include "func.c"
8. **void** main()
9. {
10. **float** Uinp[Nmax + 1], Uout[Nmax + 1];
11. **float** t0, t1, t2, tk, tmax, tmax1, U1, U2, a, b, Uinp1, C, eps, dTUMAXC;
12. **int** N, N0;
13. **char** DEBUG;
14. printf("Enter values: \nEnter tstart, t1, t2, tend:\n");
15. scanf("%f %f %f %f", &t0, &t1, &t2, &tk);
16. printf("Enter a, b:\n");
17. scanf("%f %f", &a, &b);
18. printf("Enter U1, U2:\n");
19. scanf("%f %f", &U1, &U2);
20. printf("Enter N:\n");
21. scanf("%d", &N0);
22. printf("Enter Epsilon:\n");
23. scanf("%f", &eps);
24. printf("Enter C:\n");
25. scanf("%f", &C);
26. system("cls");
27. printf("Debug mode? [Y/N]");
28. DEBUG = getch();
29. system("cls");
30. N = N0;
31. dTUMAXC = eps + 1;
32. **while** (N <= Nmax && dTUMAXC >= eps)
33. {
34. design\_t\_Uinp( N, t0, t1, t2, tk, U1, U2, Uinp);
35. design\_Uout(N,  a, b, Uinp1, Uinp, Uout);
36. tmax = design\_tmax(N, Uout, t0, tk, C, U1);
37. printf("%f", tmax);
38. **if** (DEBUG == 'y' || DEBUG == 'Y')
39. {
40. debug\_out(N, t0, tk, tmax, Uinp, Uout);
41. }
42. **if** (N != N0)
43. {
45. dTUMAXC = fabs(tmax1 - tmax) / tmax;
46. }
47. tmax1 = tmax;
48. N \*= 2;
49. }
50. **if** (N > Nmax)
51. {
52. printf("Specified accuracy is not achieved\n");
53. printf("Actual margin of error is %lf", dTUMAXC);
54. }
55. **else**
56. {
57. N = N / 2;
58. printf("Moment of time when Uout > C\*Umax: %lf\n", tmax);
59. printf("Specified accuracy is achieved at N = %d\n", N);
60. printf("Actual margin of error is %lf\n", dTUMAXC);
61. }
62. system("pause");
63. }

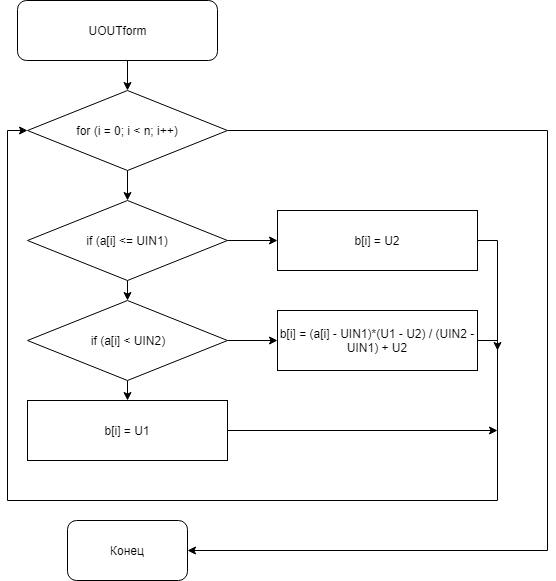
**Файл func.c:**

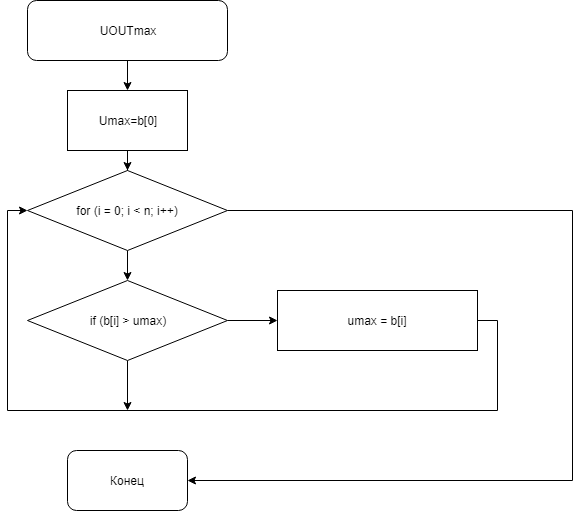
1. **void** design\_t\_Uinp(**int** N, **float** t0, **float** t1, **float** t2, **float** tk, **float** U1, **float** U2, **float** \*Uinp)
2. {
3. **float** dt, t;
4. t = t0;
5. dt = (tk - t0)/N;
6. **for** (**int** i = 0; i < N; i++)
7. {
8. **if** (t <= t1)
9. {
10. Uinp[i] =  (t - t0) \* U1/(t1 - t0);
11. }
12. **else** **if** ((t > t1) && (t <= t2))
13. {
14. Uinp[i] = (t - t1) \* (U2 - U1)/(t2 -t1) + U1;
15. }
16. **else**
17. {
18. Uinp[i] = (t -t2) \* -U2/(tk -t2) + U2;
19. }
20. t += dt;
21. }
22. }
24. **void** design\_Uout(**int** N, **float** a, **float** b, **float** Uinp1, **float** \*Uinp, **float** \*Uout)
25. {
26. Uinp1 = 10;
27. **for**(**int** i = 0; i < N; i++)
28. {
29. **if**(Uinp[i] <= Uinp1)
30. {
31. Uout[i] = a;
32. }
33. **else**
34. {
35. Uout[i] = Uinp[i] \* Uinp[i] \* b;
36. }
37. }
38. }
40. **float** design\_tmax(**int** N, **float** \*Uout, **float** t0, **float** tk, **float** C, **float** U1)
41. {
42. **int** i =1;
43. **float** dt = (tk - t0)/N;
44. **float** CUmax = C \* U1;
45. **while** (Uout[i] > CUmax || Uout[i-1] <= CUmax)
46. {
47. i++;
48. }
49. **return** (t0 + i \* dt);
50. }
52. **void** debug\_out(**int** N, **float** t0, **float** tk, **float** tmax, **float** \*Uinp, **float** \*Uout)
53. {
54. **float** dt = (tk -t0) /N;
55. **int** i = 0;
56. printf("T\tUin\tUout\n");
57. **while** (i <= N)
58. {
59. printf("%f %f %f\n", t0, Uinp[i], Uout[i]);
60. t0 += dt;
61. i ++;
62. }
63. printf("N=%d\nMoment of time when Uout < C\*Umax: %f s\n", N, tmax);
64. system("pause");
65. system("cls");
66. }

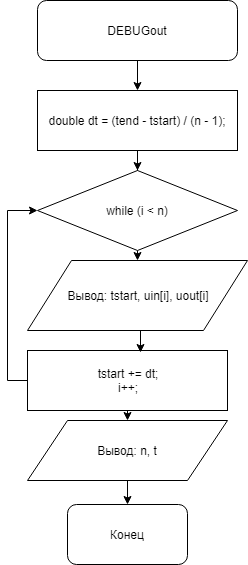
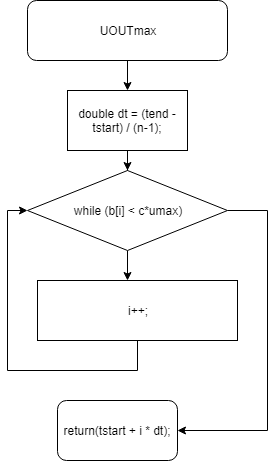
**Файл func.h:**

1. //def\_func.h--заголовочный с объявлениями пpототипов функций
3. //функция pасчета t и Uвх(t)
4. **void** design\_t\_Uinp(**int** N, **float** t0, **float** t1, **float** t2, **float** tk, **float** U1, **float** U2, **float** \*Uinp);
6. //функция pасчета Uвых(t)
7. **void** design\_Uout(**int** N, **float** a, **float** b, **float** Uinp1, **float** \*Uinp, **float** \*Uout);
8. //функция pасчета хаpактеpистики
9. **float** design\_tmax(**int** N, **float** \*Uout, **float** t0, **float** tk, **float** C, **float** U1);
11. //Функция отладочного вывода
12. **void** debug\_out(**int** N, **float** t0, **float** tk, **float** tmax, **float** \*Uinp, **float** \*Uout);
13. **Блок – схемы:**









1. Графики:

Графики построенные по формулам находятся на листе с ручными вычислениями.

Графики построенные по данным вычисленными программой:

