Титульник

$N_g = 28, N_s = 7.$

Содержание

Задание 1

Решение системы линейных уравнений с комплексными коэффициентами методом исключения Гаусса

Решить систему линейных уравнений 3-го порядка с комплексными коэффициентами методом исключения Гаусса:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1, \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2, \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3, \end{cases}$$

где $a_{11} = (N_g + 4) + j5$, $a_{12} = -3 - j4$, $a_{13} = 4 - j4$, $b_1 = 3 + j6$, $a_{21} = -3 + j2$, $a_{22} = 8 + j(10 - N_s)$, $a_{23} = 1 + j2$, $b_2 = 1 - j(N_s - 20)$, $a_{31} = j(N_g + 1)$, $a_{32} = N_s - 10$, $a_{33} = N_s - j(N_g)$, $b_3 = j10$, $N_g = 28$, $N_s = 7$.

В отчёте представить последовательность действий и промежуточные результаты. Решения (конечный результат) представить в алгебраической и показательной форме.

Произвести проверку путём подстановки найденных решений во все три уравнения системы. Вычислить невязку для каждого уравнения (разность левой и, правой части отдельно для действительной и мнимой составляющих).

Примечание. Точность решения обычно характеризуется величиной относительной ошибки решения для каждого уравнения системы:

$$\sigma(b_i) = \frac{|\Delta b_i|}{b_i} \cdot 100\%.$$

В случае, когда вещественная или мнимая часть b_i равна 0, для определения невязки необходимо вычислять и указывать в работе только абсолютную ошибку:

$$\Delta b_i = |a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n - b_i|,$$

которая и есть невязка для данного уравнения.

Решение

В соответствии с данными условия запишем совмещённую матрицу:

$$\begin{pmatrix} 32+j5 & -3-j4 & 4-j4 & 3+j6 \\ -3+j2 & 8+j3 & 1+j2 & 1+j13 \\ j29 & -3 & 7-j28 & j10 \end{pmatrix}.$$

Разделим первую строку на $(32+j5)$:

$$\begin{pmatrix} 1 & -0{,}11058-j0{,}10772 & 0{,}10296-j0{,}14109 & 0{,}12011+j0{,}168733 \\ -3+j2 & 8+j3 & 1+j2 & 1+j13 \\ j29 & -3 & 7-j28 & j10 \end{pmatrix}.$$

Умножим первую строку на $(-3\ +j2)$:

$$\begin{pmatrix} -3+j2 & 0{,}54719+j0{,}10200 & -0{,}02669+j0{,}62917 & -0{,}69781-j0{,}26597 \\ -3+j2 & 8+j3 & 1+j2 & 1+j13 \\ j29 & -3 & 7-j28 & j10 \end{pmatrix}.$$

Вычтем первую строку из второй строки и восстановим её:

$$\begin{pmatrix} 1 & -0{,}11058-j0{,}10772 & 0{,}10296-j0{,}14109 & 0{,}12011+j0{,}168733 \\ 0 & 7{,}45281+j2{,}89800 & 1{,}02669+j1{,}37083 & 1{,}69781+j13{,}26597 \\ j29 & -3 & 7-j28 & j10 \end{pmatrix}.$$

Умножим первую строку на $j29$:

$$\begin{pmatrix} j29 & 3{,}12393-j3{,}20686 & 4{,}09152+j2{,}98570 & -4{,}89323+j3{,}48332 \\ 0 & 7{,}45281+j2{,}89800 & 1{,}02669+j1{,}37083 & 1{,}69781+j13{,}26597 \\ j29 & -3 & 7-j28 & j10 \end{pmatrix}.$$

Вычтем первую строку из третьей и восстановим её:

$$\begin{pmatrix} 1 & -0{,}11058-j0{,}10772 & 0{,}10296-j0{,}14109 & 0{,}12011+j0{,}168733 \\ 0 & 7{,}45281+j2{,}89800 & 1{,}02669+j1{,}37083 & 1{,}69781+j13{,}26597 \\ 0 & -6{,}12393+j3{,}20686 & 2{,}90848-j30{,}98570 & 4{,}89323+j6{,}51668 \end{pmatrix}.$$

Получим единицу во второй строке и втором столбце разделив вторую строку на $(7{,}45281+j2{,}89800)$:

$$\begin{pmatrix} 1 & -0{,}11058-j0{,}10772 & 0{,}10296-j0{,}14109 & 0{,}12011+j0{,}168733 \\ 0 & 1 & 0{,}18179+j0{,}11324 & 0{,}79912+j1{,}46926 \\ 0 & -6{,}12393+j3{,}20686 & 2{,}90848-j30{,}98570 & 4{,}89323+j6{,}51668 \end{pmatrix}.$$

Умножим вторую строку на $(-0{,}11058-j0{,}10772)$:

$$\begin{pmatrix} 1 & -0{,}11058-j0{,}10772 & 0{,}10296-j0{,}14109 & 0{,}12011+j0{,}168733 \\ 0 & -0{,}11058-j0{,}10772 & -0{,}00790-j0{,}03211 & 0{,}06990-j0{,}24856 \\ 0 & -6{,}12393+j3{,}20686 & 2{,}90848-j30{,}98570 & 4{,}89323+j6{,}51668 \end{pmatrix}.$$

Вычтем вторую строку из первой строки и восстановим её:

$$\begin{pmatrix} 1 & 0 & 0{,}11086 - j0{,}10898 & 0{,}052021 + j0{,}41729 \\ 0 & 1 & 0{,}18179 + j0{,}11324 & 0{,}79912 + j1{,}46926 \\ 0 & -6{,}12393 + j3{,}20686 & 2{,}90848 - j30{,}98570 & 4{,}89323 + j6{,}51668 \end{pmatrix}.$$

Умножим вторую строку на $(-6{,}12393 + j3{,}20686)$:

$$\begin{pmatrix} 1 & 0 & 0{,}11086 - j0{,}10898 & 0{,}052021 + j0{,}41729 \\ 0 & -6{,}12393 + j3{,}20686 & -1{,}47645 - j0{,}11051 & -9{,}60549 - j6{,}43495 \\ 0 & -6{,}12393 + j3{,}20686 & 2{,}90848 - j30{,}98570 & 4{,}89323 + j6{,}51668 \end{pmatrix}.$$

Вычтем вторую строку из третьей и восстановим вторую строку:

$$\begin{pmatrix} 1 & 0 & 0{,}11086 - j0{,}10898 & 0{,}052021 + j0{,}41729 \\ 0 & 1 & 0{,}18179 + j0{,}11324 & 0{,}79912 + j1{,}46926 \\ 0 & 0 & 4{,}38494 - j30{,}87519 & 14{,}49872 + j12{,}95164 \end{pmatrix}.$$

Разделим элементы третьей строки на $(4{,}38494 - j30{,}87519)$:

$$\begin{pmatrix} 1 & 0 & 0{,}11086 - j0{,}10898 & 0{,}052021 + j0{,}41729 \\ 0 & 1 & 0{,}18179 + j0{,}11324 & 0{,}79912 + j1{,}46926 \\ 0 & 0 & 1 & -0{,}34582 + j0{,}51870 \end{pmatrix}.$$

Умножим третью строку на $(0{,}11086 - j0{,}10898)$:

$$\begin{pmatrix} 1 & 0 & 0{,}11086 - j0{,}10898 & 0{,}052021 + j0{,}41729 \\ 0 & 1 & 0{,}18179 + j0{,}11324 & 0{,}79912 + j1{,}46926 \\ 0 & 0 & 0{,}11086 - j0{,}10898 & 0{,}01819 + j0{,}09519 \end{pmatrix}.$$

Вычтем третью строку из первой строки и восстановим третью строку:

$$\begin{pmatrix} 1 & 0 & 0 & 0{,}03202 + j0{,}32210 \\ 0 & 1 & 0{,}18179 + j0{,}11324 & 0{,}79912 + j1{,}46926 \\ 0 & 0 & 1 & -0{,}34582 + j0{,}51870 \end{pmatrix}.$$

Умножим третью строку на $(0{,}18179 + j0{,}11324)$:

$$\begin{pmatrix} 1 & 0 & 0 & 0{,}03202 + j0{,}32210 \\ 0 & 1 & 0{,}18179 + j0{,}11324 & 0{,}79912 + j1{,}46926 \\ 0 & 0 & 0{,}18179 + j0{,}11324 & -0{,}12161 + j0{,}05514 \end{pmatrix}.$$

Вычтем третью строку из второй строки и восстановим третью строку:

$$\begin{pmatrix} 1 & 0 & 0 & 0{,}03202 + j0{,}32210 \\ 0 & 1 & 0 & 0{,}92073 + j1{,}41423 \\ 0 & 0 & 1 & -0{,}34582 + j0{,}51870 \end{pmatrix}.$$

Т.е. получили следующее решение системы алгебраических линейных уравнений с комплексными коэффициентами в алгебраической форме:

$$x_1 = 0{,}03202 + j0{,}322109,$$

$$x_2 = 0{,}92073 + j1{,}41423,$$

$$x_3 = -0{,}34582 + j0{,}51870.$$

Запишем результаты решения в показательной форме:

$$x_1 = 0{,}3237 \cdot e^{j84{,}3230^\circ},$$

$$x_2 = 1{,}6875 \cdot e^{j56{,}9340^\circ},$$

$$x_3 = 0{,}6234 \cdot e^{j123{,}6916^\circ}.$$

Рассчитываем невязки при подстановке полученного решения СЛАУ в каждое уравнение:

$$(32 + j5) \cdot (0{,}03202 + j0{,}322109) + (-3 - j4) \cdot (0{,}92073 + j1{,}41423) + (4 - j4) \cdot (-0{,}34582 + j0{,}51870) = 3 + j6.$$

Невязка для действительной части:

$$|3 - 3{,}001| = 0{,}001.$$

Невязка для мнимой части:

$$|6 - 6{,}001| = 0{,}001.$$

$$(-3 + j2) \cdot (0{,}03202 + j0{,}322109) + (8 + j3) \cdot (0{,}92073 + j1{,}41423) + (1 + j2) \cdot (-0{,}34582 + j0{,}51870) = 1 + j13.$$

Невязка для действительной части:

$$|1 - 1{,}002| = 0{,}002.$$

Невязка для мнимой части:

$$|13 - 13{,}002| = 0{,}002.$$

$$j29 \cdot (0{,}03202 + j0{,}322109) - 3 \cdot (0{,}92073 + j1{,}41423) + (7 - j28) \cdot (-0{,}34582 + j0{,}51870) = j10.$$

Невязка для действительной части:

$$|0 - 0{,}001| = 0{,}001.$$

Невязка для мнимой части:

$$|10 - 10| = 0.$$

Задание 2

Методы приближения функции. Интерполяция и аппроксимация

Задана функция одной переменной $y(x)$ в виде таблицы значений:

x	-1	28	6	10
$y(x)$	1	2	8	-2

Интерполировать функцию $y(x)$ полиномом Лагранжа 3-го порядка $L_3(x)$. Выполнить проверку правильности интерполяции по всем точкам.

Аппроксимировать функцию $y(x)$ по методу наименьших квадратов полиномом 2-го порядка $\varphi_2(x)$.

Построить графики интерполяции $L_3(x)$ и аппроксимации $\varphi_2(x)$ на одном рисунке в интервале $x \in [x_{min}, x_{max}]$ из таблицы и отметить на поле графика заданные табличные точки.

Решение

1.Интерполяция

Строим полином Лагранжа $L_3^r(x)$ по этим узлам.

Общая формула:

$$L_n(x) = \sum_{i=0}^{n} P_{ni}(x) \cdot f_i,$$

где

$$P_{ni}(x) = \frac{(x - x_0) \cdot \ldots \cdot (x - x_{i-1}) \cdot (x - x_{i+1}) \cdot \ldots \cdot (x - x_n)}{(x_i - x_0) \cdot \ldots \cdot (x_i - x_{i-1}) \cdot (x_i - x_{i+1}) \cdot \ldots \cdot (x_i - x_n)}.$$

Для 4-х точек:

$$L_3(x) = \frac{(x-x_1)\cdot(x-x_2)\cdot(x-x_3)}{(x_0-x_1)\cdot(x_0-x_2)\cdot(x_0-x_3)}\cdot f_0 + \frac{(x-x_0)\cdot(x-x_2)\cdot(x-x_3)}{(x_1-x_0)\cdot(x_1-x_2)\cdot(x_1-x_3)}$$

$$\cdot f_1 + \frac{(x-x_0)\cdot(x-x_1)\cdot(x-x_3)}{(x_2-x_0)\cdot(x_2-x_1)\cdot(x_2-x_3)}\cdot f_2$$

$$+ \frac{(x-x_0)\cdot(x-x_1)\cdot(x-x_2)}{(x_3-x_0)\cdot(x_3-x_1)\cdot(x_3-x_2)}\cdot f_3$$

$$= \frac{(x-28)\cdot(x-6)\cdot(x-10)}{(-1-28)\cdot(-1-6)\cdot(-1-10)}\cdot 1$$

$$+ \frac{\big(x-(-1)\big)\cdot(x-6)\cdot(x-10)}{\big(28-(-1)\big)\cdot(28-6)\cdot(28-10)}\cdot 2$$

$$+ \frac{\big(x-(-1)\big)\cdot(x-28)\cdot(x-10)}{\big(6-(-1)\big)\cdot(6-28)\cdot(6-10)}\cdot 8$$

$$- \frac{\big(x-(-1)\big)\cdot(x-28)\cdot(x-6)}{\big(10-(-1)\big)\cdot(10-28)\cdot(10-6)}\cdot 2$$

$$= \frac{175}{11484}\cdot x^3 - \frac{2093}{3828}\cdot x^2 + \frac{18727}{5742}\cdot x + \frac{4616}{957}.$$

Т.е. получили значение функции полинома Лагранжа:

$$L_3(x) = \frac{175}{11484}\cdot x^3 - \frac{2093}{3828}\cdot x^2 + \frac{18727}{5742}\cdot x + \frac{4616}{957}$$

$$= 0{,}01524\cdot x^3 - 0{,}54676\cdot x^2 + 3{,}26141\cdot x + 4{,}82341.$$

Проверка:

$$L_3(-1) = 0{,}01524\cdot(-1)^3 - 0{,}54676\cdot(-1)^2 + 3{,}26141\cdot(-1) + 4{,}82341$$

$$= 1,$$

$$L_3(28) = 0{,}01524\cdot 28^3 - 0{,}54676\cdot 28^2 + 3{,}26141\cdot 28 + 4{,}82341 = 2,$$

$$L_3(6) = 0{,}01524\cdot 6^3 - 0{,}54676\cdot 6^2 + 3{,}26141\cdot 6 + 4{,}82341 = 8,$$

$$L_3(10) = 0{,}01524\cdot 10^3 - 0{,}54676\cdot 10^2 + 3{,}26141\cdot 10 + 4{,}82341 = -2.$$

2. Аппроксимация

Полином 2-го порядка имеет вид:

$$\varphi_2(x) = a_0 + a_1 x + a_2 x^2.$$

Коэффициенты при переменной x можно найти, решив систему уравнений.

$$\begin{cases} a_0 N + a_1 \sum_{k=1}^{N} x_k + a_2 \sum_{k=1}^{N} x_k^2 = \sum_{k=1}^{N} y_k \\ a_0 \sum_{k=1}^{N} x_k + a_1 \sum_{k=1}^{N} x_k^2 + a_2 \sum_{k=1}^{N} x_k^3 = \sum_{k=1}^{N} x_k y_k \\ a_0 \sum_{k=1}^{N} x_k^2 + a_1 \sum_{k=1}^{N} x_k^3 + a_2 \sum_{k=1}^{N} x_k^4 = \sum_{k=1}^{N} x_k^2 y_k \end{cases}$$

Для составления системы найдём значения коэффициентов при неизвестных. Для этого составим вспомогательную таблицу:

	x	y	x^2	x^3	x^4	xy	x^2y
	-1	1	1	-1	1	-1	1
	6	8	36	216	1296	48	288
	10	-2	100	1000	10000	-20	-200
	28	2	784	21952	614656	56	1568
Сумма	**43**	**9**	**921**	**23167**	**625953**	**83**	**1657**

Получим систему:

$$\begin{cases} 4a_0 + 43a_1 + 921a_2 = 9 \\ 43a_0 + 921a_1 + 23167a_2 = 83 \\ 921a_0 + 23167a_1 + 625953a_2 = 1657 \end{cases}$$

Решим её методом Крамера:

$$\Delta = \begin{vmatrix} 4 & 43 & 921 \\ 43 & 921 & 23167 \\ 921 & 23167 & 625953 \end{vmatrix} = 55516940,$$

$$\Delta_{a_0} = \begin{vmatrix} 9 & 43 & 921 \\ 83 & 921 & 23167 \\ 1657 & 23167 & 625953 \end{vmatrix} = 140200720,$$

$$\Delta_{a_1} = \begin{vmatrix} 4 & 9 & 921 \\ 43 & 83 & 23167 \\ 921 & 1657 & 625953 \end{vmatrix} = -728860,$$

$$\Delta_{a_2} = \begin{vmatrix} 4 & 43 & 9 \\ 43 & 921 & 83 \\ 921 & 23167 & 1657 \end{vmatrix} = -32340.$$

Вычисляем значения коэффициентов квадратного полинома:

$$a_0 = \frac{\Delta_{a_0}}{\Delta} = \frac{140200720}{55516940} = 2{,}5252,$$

$$a_1 = \frac{\Delta_{a_1}}{\Delta} = \frac{-728860}{55516940} = -0{,}0131,$$

$$a_2 = \frac{\Delta_{a_2}}{\Delta} = \frac{-32340}{55516940} = -0{,}0006.$$

Таким образом. полином 2-го порядка имеет вид:

$$\varphi_2(x) = 2{,}5252 - 0{,}0131x - 0{,}0006x^2.$$

Проверим правильность решения системы:

$$\varphi_2(-1) = 2{,}5252 - 0{,}0131 \cdot (-1) - 0{,}0006 \cdot (-1)^2 = 2{,}5378,$$

$$\varphi_2(6) = 2{,}5252 - 0{,}0131 \cdot 6 - 0{,}0006 \cdot 6^2 = 2{,}4255,$$

$$\varphi_2(10) = 2{,}5252 - 0{,}0131 \cdot 10 - 0{,}0006 \cdot 10^2 = 2{,}3357,$$

$$\varphi_2(28) = 2{,}5252 - 0{,}0131 \cdot 28 - 0{,}0006 \cdot 28^2 = 1{,}7010.$$

Строим графики интерполяционного полинома и квадратичного полинома (рис. 1). На рис. 1 исходные данные представлены точками.

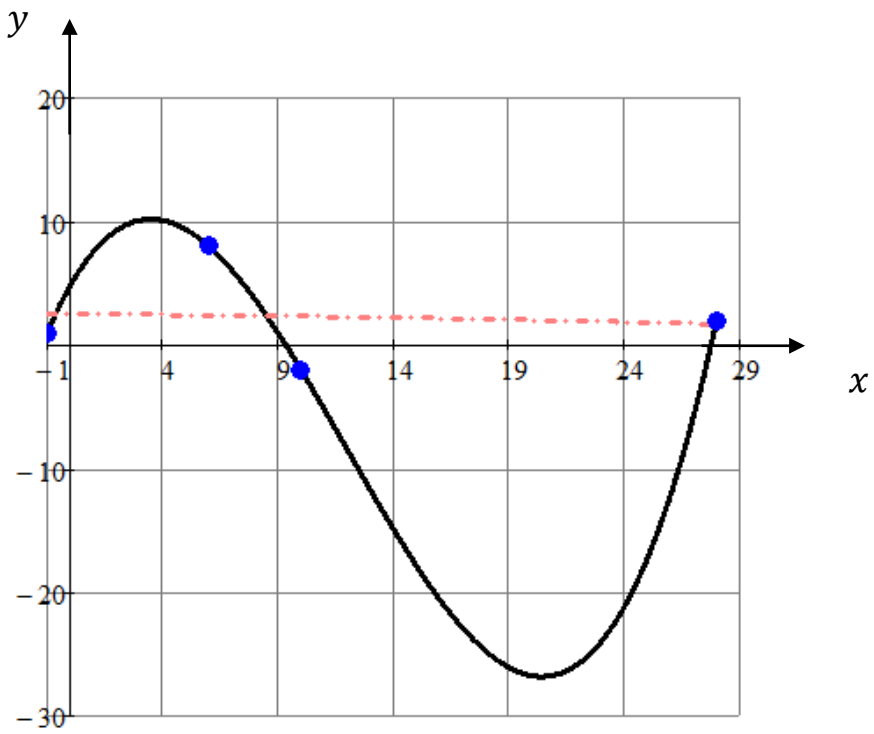

Рис. 1 – Графики интерполяционного полинома (сплошная линия) и аппроксимирующего квадратичного полинома (штрих-пунктирная линия)

Как видно из графика, полином Лагранжа 3-й степени значительно точнее квадратичного полинома аппроксимации.

Задание 3

Методы численного интегрирования и дифференцирования функции

Задана функция одной переменной $f(x)$ и границы интервала a и b (выбрать из таблицы в соответствии с номером по журналу N_s). $N_s = 7$

Номер варианта	Вид функции $f(x)$	Границы интервала	
		a	b
7	$x \cdot 2^{3x}$	1	4

Вычислить определённый интеграл

$$I = \int_a^b f(x)dx$$

на интервале $[a,b]$, разделяя интервал на $n = 5$ частей с шагом $h = (b-a)/n$:

- методом прямоугольников;
- методом трапеций;
- методом Симпсона.

Сравнить полученные результаты.

Вычислить производную по методу центральных разностей $f'(x)$ и интеграл с переменным верхним пределом

$$F(x) = \int_a^x f(x)dx$$

по методу трапеций, выбирая шаг h. Результаты занести в таблицу.

Выбрав соответствующие масштабы, построить графики функций $f(x)$, $f'(x)$ и $F(x)$ на одном рисунке в интервале $x \in [a,b]$.

Решение

1. Вычислим определённый интеграл

$$I = \int_1^4 x \cdot 2^{3x} dx$$

С количеством интервалов $n = 5$ и шагом

$$h = \frac{4-1}{5} = 0{,}6.$$

1) Метод прямоугольников:

$$\int_a^b f(x)dx \approx f\left(\frac{a+b}{2}\right)(b-a)$$

i	$x_i = \frac{a+b}{2}$	$f(x_i)$
1	$x_1 = 1{,}3$	19,40709
2	$x_2 = 1{,}9$	98,76989
3	$x_3 = 2{,}5$	452,54834
4	$x_4 = 3{,}1$	1954,07241
5	$x_5 = 3{,}7$	8121,47059
	Сумма	10646,26832

$$I = \int_1^4 x \cdot 2^{3x} dx \approx h \sum_{i=1}^{5} f(x_i)$$

$$I = 0{,}6 \cdot 10646{,}26832 = 6387{,}76099.$$

2) Метод трапеций:

$$\int_a^b f(x)dx \approx \frac{f(a)+f(b)}{2}(b-a)$$

i	$x_i = i \cdot h$	$f(x_i)$
0	$x_0 = 1$	8
1	$x_1 = 1{,}6$	44,57219
2	$x_2 = 2{,}2$	213,41289
3	$x_3 = 2{,}8$	945,82327
4	$x_4 = 3{,}4$	3999,30819
5	$x_5 = 4$	16384

$$I = \int_1^4 x \cdot 2^{3x} \approx h\left(\frac{f(x_0)+f(x_5)}{2} + \sum_{i=1}^{4} f(x_i)\right)$$

$$I = 0{,}6 \cdot \left(\frac{8 + 16384}{2} + 44{,}57219 + 213{,}41289 + 945{,}82327 + 3999{,}30819\right)$$
$$= 8039{,}46993.$$

3) Метод Симпсона:

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + 2f(b)\right)$$

i	$x_i = i \cdot \frac{h}{2}$	$f(x_i)$
0	$x_0 = 1$	8
1	$x_1 = 1{,}3$	19,40709
2	$x_2 = 1{,}6$	44,57219
3	$x_3 = 1{,}9$	98,76989
4	$x_4 = 2{,}2$	213,41289
5	$x_5 = 2{,}5$	452,54834
6	$x_6 = 2{,}8$	945,82327
7	$x_7 = 3{,}1$	1954,07241
8	$x_8 = 3{,}4$	3999,30819
9	$x_9 = 3{,}7$	8121,47059
10	$x_{10} = 4$	16384

$$I = \int_1^4 x \cdot 2^{3x}$$
$$\approx \frac{0{,}6}{6}$$
$$\cdot \big(8 + 16384 + 4$$
$$\cdot (19{,}40709 + 98{,}76989 + 452{,}54834 + 1954{,}07241$$
$$+ 8121{,}47059) + 2$$
$$\cdot (44{,}57219 + 213{,}41289 + 945{,}82327 + 3999{,}30819)\big)$$
$$= 6938{,}33064$$

2. Сравним полученные результаты

Вычислим аналитически точное решение:

$$I = \int_1^4 x \cdot 2^{3x} \approx 6929{,}78729.$$

Сравним ответы различных методов и их абсолютные погрешности для данного примера:

	Метод прямоугольников	Метод трапеций	Метод Симпсона
Результат	6387,76099	8039,46993	6938,33064
Погрешность	0,078217	0,1601323	0,001233

Ответ, близкий к точному, дал метод Симпсона. Это правдоподобно, поскольку метод использует более сложную аппроксимацию, чем метод прямоугольников и трапеций. Метод трапеций оказался менее точным, чем метод прямоугольников, что возможно, однако метод трапеций обычно точнее, потому что использует линейную интерполяцию вместо константной, как у метода прямоугольников.

3. Вычислим производную по точкам методом центральных разностей:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

i	$x_i = i \cdot h + \frac{h}{2}$	$f(x_i - \frac{h}{2})$	$f(x_i + \frac{h}{2})$	$f'(x_i) \approx \frac{f\left(x_i + \frac{h}{2}\right) - f(x_i - \frac{h}{2})}{h}$
0	1,4	0	0,19194	0,11996
1	2,2	0,13288	0,35115	0,13642
2	3	0,26853	0,44048	0,10747
3	3,8	0,35658	0,49035	0,08361
4	4,6	0,41189	0,51806	0,06635

Вычислим первообразную методом трапеций, используя предыдущие расчёты, полагая, что $F(1) = 0$:

$$\int_a^b f(x)dx \approx \frac{f(a) + f(b)}{2}(b - a)$$

i	$x_i = i \cdot h$	$f(x_i)$	$\frac{f(x_{i-1}) + f(x_i)}{2}h$	$F(x_i)$
0	1	8		0
1	1,6	44,57219	66,42386777	66,42386777
2	2,2	213,41289	297,1186378	363,5425056
3	2,8	945,82327	1263,816326	1627,358831
4	3,4	3999,30819	5198,946981	6826,305812
5	4	16384	1199,792458	8026,09827

Построим графики

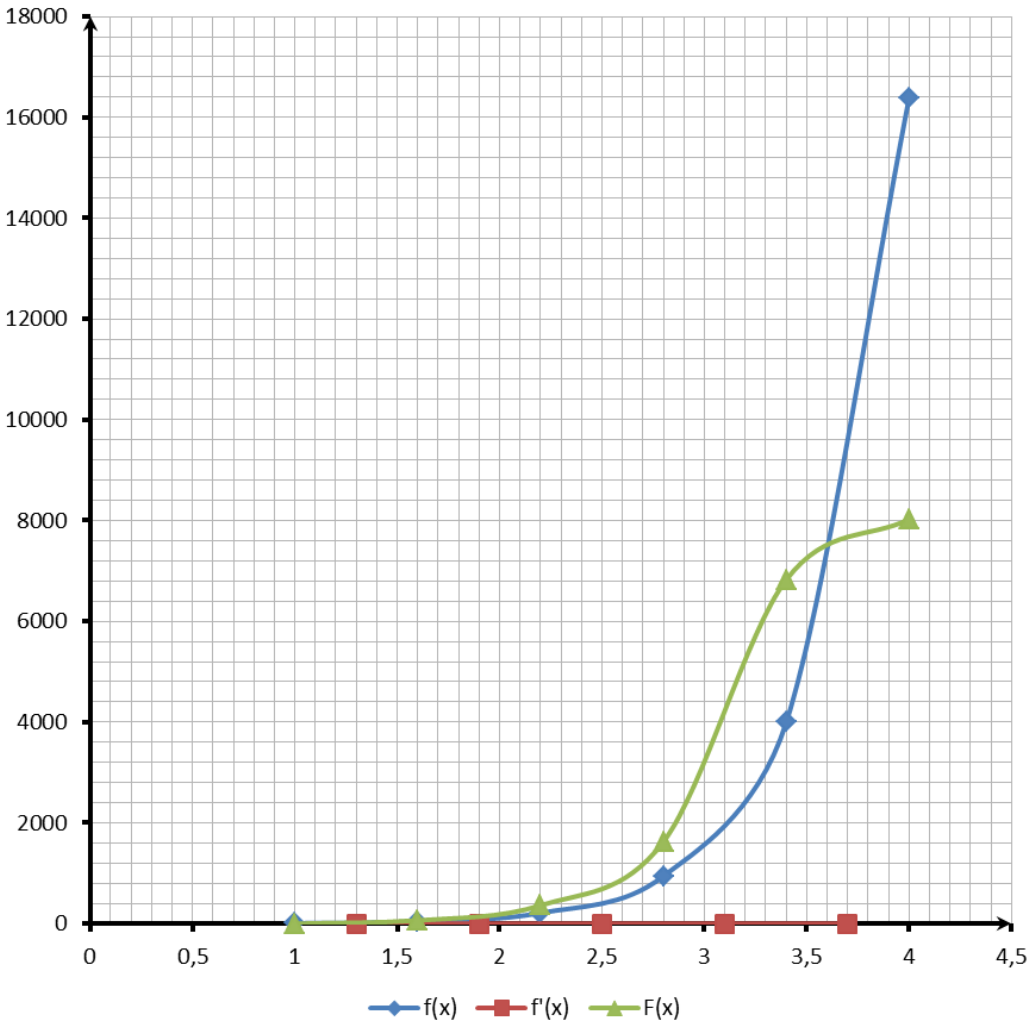

Рис. 2 – Графики подынтегральной функции, производной и первообразной

Задание 4

Решение нелинейного уравнения

Решить нелинейное уравнение $f(x) = N_g$, где $N_g = 28$, а $f(x)$ – функция из предыдущего задания, четырьмя различными методами:

- методом бисекции;
- методом хорд;
- методом Ньютона;
- методом простых итераций (последовательных приближений).

Выполнить по 6 итераций каждым методом, сравнить погрешность вычислений.

Примечание. Начальное приближение (границы интервала) следует выбирать так, чтобы на интервале находился только один корень уравнения. Если полученное уравнение не имеет решения, согласовать с преподавателем изменение исходных данных.

Решение

Условие задания:

$$x \cdot 2^{3x} = 28.$$

Для нахождения графическим способом интервала нахождения корня создаём m-файлы:

Листинг 1. Файл-функция fun.m

```
function y=fun(x)
y=x.*2.^(3*x)-28;
end
```

Листинг 2. Файл-сценарий grafic.m

```
% Построение графика функции
a=0.1;
b=2;
x=a:0.01:b;
y=fun(x);
plot(x,y);
grid on
ylabel('y') % название оси y
xlabel('x') % название оси x
title('График функции')
```

После запуска файла grafic.m в графическом окне появляется построенный график функции на заданном по умолчанию (задание 3) интервале (рис. 3).

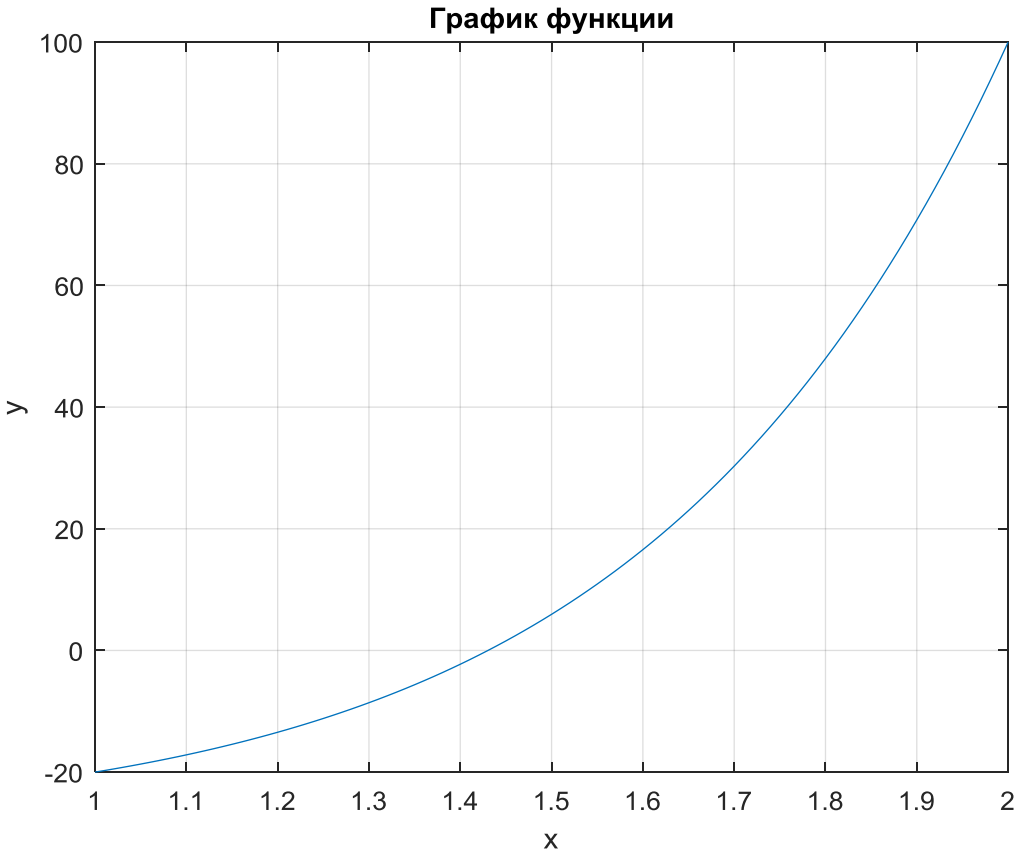

Рис. 3 – График функции на заданном интервале

Как можно заметить по графику рис. 1 корень приблизительно находится на интервале $x \in [1{,}3; 1{,}6]$.

Погрешность поиска значения корня на каждой итерации определяется по формуле:

$$|x_{i+1} - x_i| = \varepsilon$$

1. Метод бисекции

Метод бисекции (метод деления отрезка пополам) заключается в том, что промежуток, на котором функция имеет корень, последовательно делится пополам до достижении заданной точности.

На первом шаге данного алгоритма осуществляется проверка знаков функции на концах промежутка, если соблюдается условие f(a)*f(b)<0, то функция пересекает ось Ox, а значит, имеет корень на данном промежутке.

Далее промежуток [a, b] делится пополам:

$$x_1 = \frac{a+b}{2}$$

Вычисляется значение функции в точке x1 и сравнивается со значениями функции в точках a и b, если f(x1)*f(a)<0, то b = x1, если f(x1)*f(b)<0, то a = x1. Получаем промежуток в два раза меньше исходного, который также делим пополам и сравниваем знаки функций в найденных точках. Осуществляя вышеуказанные операции определённое количество раз, делаем промежуток, на котором определяется корень функции все меньше и меньше, добиваясь требуемой точности.

Для решения поставленной задачи создаём m-файл, представленный на листинге 3.

Листинг 3. Файл-сценарий bicekciya.m

```
%скрипт-файл поиска приближенного значения корня
уравнения методом бисекции
clear; % очистка памяти
clc;   % очистка командного окна
a = 1.3; %инициализация переменной значением начала
интервала
b = 1.6 %инициализация переменной значением конца
интервала
x = 0; %переменная, в которой будут храниться середины
интервалов
x_prev = a;%переменная для вычисления погрешности
найденного значения
%корня на каждой итерации
func_bin = @(x)x*2^(3*x)-28;%присваивание переменной
дескриптора функции
%цикл поиска значения корня, состоит из 6 шагов
for i = 1:6
    x = (a+b)/2;%присваивание переменной значения
середины интервала
    disp(['Приближенное значение корня на ', num2str(i)
,  ' итерации']);
    %вывод на экран приближенного значения корня
      disp(x);
      disp('Погрешность');
      %вывод на экран погрешности
       disp(abs(x-x_prev));
```

```
        disp('Значение функции');
        %вывод на экран значения функции, полученного на
i итерации f(x)
     disp(feval(func_bin, x));
     %условный оператор осуществляет проверку условия
f(x)*f(a)<0
    if  feval(func_bin, x)*feval(func_bin, a)<0
        %условие истинно. Значение  переменной 'x'
присваивается переменной
        %'b'. Текущее значение переменной 'x'
присваивается переменной 'x_prev'
        b = x;
        x_prev = x;
    else
        %условие ложно. Значение переменной 'x'
присваивается переменной
        %'a'. Текущее значение переменной 'x'
присваивается переменной 'x_prev'
        a = x;
        x_prev = x;
    end
end
```

После запуска файла bicekciya.m на выполнение получаем следующие результаты в командном окне:

```
Приближенное значение корня на 1 итерации
    1.4500

Погрешность
    0.1500

Значение функции
    1.5698

Приближенное значение корня на 2 итерации
    1.3750

Погрешность
    0.0750

Значение функции
   -4.0088

Приближенное значение корня на 3 итерации
    1.4125

Погрешность
    0.0375
```

```
Значение функции
   -1.3558

Приближенное значение корня на 4 итерации
    1.4313

Погрешность
    0.0188

Значение функции
    0.0713

Приближенное значение корня на 5 итерации
    1.4219

Погрешность
    0.0094

Значение функции
   -0.6509

Приближенное значение корня на 6 итерации
    1.4266

Погрешность
    0.0047

Значение функции
   -0.2920
```

На шестой итерации получаем приближенное значение корня x = 1,4266, с погрешностью 0,0047. Значение функции равно -0,2920.

2. Метод хорд

Определяя корень на некотором промежутке, выбираем две точки на графике функции A1(x1;y1) и A2(x2; y2) и проведём через них прямую, она пересекает ось Ox в точке (x3; 0). Найдём на графике функции точку A3(x3; y3) и вместо точек A1 и A2 возьмём точки A3 и A2, проведём через них прямую и найдём её точку пересечения с осью Ox. Продолжая эти действия, мы приближаем точки A_i и A_{i-1} друг к другу, тем самым уменьшая промежуток, на котором ведётся поиск корня.

Итерационная формула метода хорд имеет вид:

$$x_{i+1} = a - \frac{(b-a)}{f(b)-f(a)} f(a)$$

Для решения поставленной задачи создаём m-файл, представленный на листинге 4.

Листинг 4. Файл-сценарий horda.m

```
%скрипт-файл поиска корня уравнения методом хорд
clear; % очистка памяти
clc;   % очистка командного окна
a = 1.3;%инициализация переменной значением начала
интервала
b = 1.6;%инициализация переменной значением конца
интервала
x0 = 0;%инициализация итерационной переменной, в
которой будет хранится
%значение корня
x_prev = a;%переменная для вычисления погрешности
найденного значения
%корня на каждой итерации
func_hd = @(x)x*2^(3*x)-28;%присваивание переменной
дескриптора функции
%цикл поиска значения корня (6 шагов)
for i =1:6
    % итерационное уравнение метода хорд
    %x0 = a - ((b-a)/(f(b) - f(a)))*f(a)
    x0 = a - ((b - a)/(feval(func_hd, b) -
feval(func_hd, a)))*feval(func_hd, a);
    disp(['Приближенное значение корня на ',
num2str(i) ,  ' итерации']);
    %вывод на экран приближенного значения корня
    disp(x0);
    disp('Погрешность');
    %вывод на экран погрешности
    disp(abs(x0-x_prev));
    disp('Значение функции');
      %вывод на экран значения функции, полученного
на i итерации
    disp(feval(func_hd, x0));
     %условный оператор осуществляет проверку
условия f(x0)*f(a)<0
    if feval(func_hd, x0)*feval(func_hd, a)<0
        %условие истинно. Переменной 'b'
присваивается значение переменной
```

```
            %'x0'. Текущее значение переменной 'x0'
присваивается переменной 'x_prev'
            b = x0;
            x_prev = x0;
        else
            %условие ложно. Переменной 'a'
присваивается значение переменной
            %'x0'. Текущее значение переменной 'x0'
присваивается переменной 'x_prev'
            a = x0;
            x_prev = x0;
        end
    end
```

После запуска файла horda.m на выполнение получаем следующие результаты в командном окне:

```
Приближенное значение корня на 1 итерации
    1.4024

Погрешность
    0.1024

Значение функции
   -2.0933

Приближенное значение корня на 2 итерации
    1.4246

Погрешность
    0.0222

Значение функции
   -0.4432

Приближенное значение корня на 3 итерации
    1.4292

Погрешность
    0.0046

Значение функции
   -0.0909

Приближенное значение корня на 4 итерации
    1.4301

Погрешность
   9.3227e-04
```

```
Значение функции
   -0.0185

Приближенное значение корня на 5 итерации
    1.4303

Погрешность
   1.8981e-04

Значение функции
   -0.0038

Приближенное значение корня на 6 итерации
    1.4303

Погрешность
   3.8627e-05

Значение функции
  -7.6773e-04
```

На шестой итерации получаем значение корня x = 1,4303 с погрешностью 0,000038627. Значение функции равно -0,00076773.

3. Метод Ньютона

Суть метода заключается в том, что берётся начальная точка, к графику функции в этой точке строится касательная, для которой определяется точка пересечения с осью Ox, по ней определяется точка на графике функции и к этой точке строится касательная. И так далее, пока не будет достигнута требуемая точность.

Начальное значение выбирается по условию:

$$f(x) * f''(x) > 0$$

Итерационная формула Ньютона имеет вид:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Для решения поставленной задачи создаём m-файл, содержание которого отражено в листинге 5.

Листинг 5. Файл-сценарий newton.m

```
    %скрипт-файл для поиска приближенного значения
корня уравнения методом
    %Ньютона
    clear; % очистка памяти
    clc;   % очистка командного окна
    syms x; %инструкция для работы с мат. объектами в
символьном виде
    a = 1.3;%инициализация переменной значением начала
интервала
    b = 1.6;%инициализация переменной значением конца
интервала
    x0=0;%инициализация итерационной переменной
    x1 = 0;%инициализация итерационной переменной
    func_nut = @(x)x*2^(3*x)-28; % присваивание
переменной дескриптора функции
    dfunc_nut = @(x)8^x*(3*x*log(2)+1);%присваивание
переменной дескриптора
    % производной функции
    ddfunc_nut = @(x)3*8^x*log(2)*(3*x*log(2)+2); %
присваивание переменной дескриптора
    %второй производной функции
    %проверка условия f(a)*f''(a)>0
    if feval(func_nut, a)*feval(ddfunc_nut, a)>0
        %условие истинно. Выбираем в качестве начальной
точки начало интервала
        x0 = a;
    else
          %условие ложно. В качестве начальной точки
выбирается конец интервала
        x0 = b;
    end
    %цикл поиска значения корня, состоит из 6 шагов
    for i=1:6
        %итерационная формула Ньютона x(i) = x(i-1) -
f(x(i-1))/f'(x(i-1))
      x1 = x0 - (feval(func_nut, x0))/(feval(dfunc_nut,
x0));
       disp(['Приближенное значение корня на ',
num2str(i), ' итерации']);
       %вывод на экран приближенного значения корня
      disp(x1);
     disp('Погрешность');
      %вывод на экран погрешности
     disp(abs(x1-x0));
     disp('Значение функции');
```

```
      %вывод на экран значения функции, полученного на
i итерации f(x1)
      disp(feval(func_nut, x1));
      x0 = x1;%изменение значения переменной итерации x0
     end
```

После запуска файла newton.m на выполнение получаем следующие результаты в командном окне:

```
Приближенное значение корня на 1 итерации
    1.4625

Погрешность
    0.1375

Значение функции
    2.6118

Приближенное значение корня на 2 итерации
    1.4316

Погрешность
    0.0309

Значение функции
    0.1020

Приближенное значение корня на 3 итерации
    1.4303

Погрешность
    0.0013

Значение функции
   1.7311e-04

Приближенное значение корня на 4 итерации
    1.4303

Погрешность
   2.2250e-06

Значение функции
   5.0122e-10

Приближенное значение корня на 5 итерации
    1.4303

Погрешность
   6.4424e-12
```

```
Значение функции
  -1.0658e-14

Приближенное значение корня на 6 итерации
    1.4303

Погрешность
   2.2204e-16

Значение функции
   1.0658e-14
```

На шестой итерации получаем значение корня x = 1,4303, с погрешностью 0,00000000000000022204. Значение функции равно 0,000000000000010658.

4. Метод простых итераций

Идея метода заключается в том, чтобы уравнение f(x) = 0 привести к эквивалентному уравнению

$$x = x - \mu(x)f(x)$$

Итерационная формула имеет вид:

$$x_{i+1} = x_i - \mu f(x_i)$$

Для обеспечения сходимости данного алгоритма, необходимо, чтобы значение производной функции на отрезке было по модулю меньше единицы. Найдём значение производной функции:

$$f'(x) = 8^x \cdot (3 \cdot x \cdot \ln 2 + 1)$$

Своё максимальное значение по модулю она принимает в точке x = 1,6

$$f'(1{,}6) = 8^{1{,}6} \cdot (3 \cdot 1{,}6 \cdot \ln 2 + 1) = 120{,}543.$$

Разделим обе части исходного уравнения на 10500:

$$\frac{x \cdot 2^{3 \cdot x}}{120{,}543} - \frac{28}{120{,}543} = 0$$

Составим итерационную формулу:

$$x_{i+1} = x_i - \frac{x \cdot 2^{3 \cdot x}}{120{,}543} + \frac{28}{120{,}543}.$$

Для решения поставленной задачи создаём m-файл, содержание которого отражено в листинге 6.

Листинг 6. Файл-сценарий iteraciya.m

```
%скрипт-файл для поиска приближенного значения корня
уравнения методом
%простых итераций
clear; % очистка памяти
clc;   % очистка командного окна
a = 1.3; %инициализация переменной значением начала
интервала
b = 1.6; %инициализация переменной значением конца
интервала
fmax = 0;%переменная для хранения максимального
значения производной функции
x1 = 0; %инициализация итерационной переменной
func = @(x)x*2^(3*x)-28;%присваивание переменной
дескриптора функции
dfunc = @(x)8^x*(3*x*log(2)+1); % присваивание
переменной дескриптора
% производной функции, умноженной на -1
[x, fmax] = fminbnd(dfunc, a, b);%вычисление
максимального значения производной на
%интервале [a, b]
func_it = @(x)(x*2^(3*x)-28)/fmax;%присваивание
переменной дескриптора функции
%имеющей вид, необходимый для применения к ней метода
простых итераций
x0 = (a+b)/2;%присваивание итерационной переменой
значения начала интервала
%цикл поиска значения корня, состоит из 6 шагов
for i=1:6
    %итерационная функция x(i) = x(i-1) - func_it(x0)
   x1 = x0 - feval(func_it, x0);
   disp(['Приближенное значение корня на ', num2str(i),
' итерации']);
    %вывод на экран приближенного значения корня
   disp(x1);
    disp('Погрешность');
     %вывод на экран погрешности
    disp(abs(x1-x0));
     disp('Значение функции');
      %вывод на экран значения функции, полученного на
i итерации f(x1)
     disp(feval(func, x1));
```

```
    x0 = x1;%изменение значения переменной итерации x0
end
```

После запуска файла iteraciya.m на выполнение получаем следующие результаты в командном окне:

```
Приближенное значение корня на 1 итерации
    1.4216

Погрешность
    0.0284

Значение функции
   -0.6712

Приближенное значение корня на 2 итерации
    1.4337

Погрешность
    0.0121

Значение функции
    0.2668

Приближенное значение корня на 3 итерации
    1.4289

Погрешность
    0.0048

Значение функции
   -0.1096

Приближенное значение корня на 4 итерации
    1.4309

Погрешность
    0.0020

Значение функции
    0.0444

Приближенное значение корня на 5 итерации
    1.4301

Погрешность
   8.0371e-04

Значение функции
   -0.0181

Приближенное значение корня на 6 итерации
```

```
    1.4304

Погрешность
   3.2767e-04

Значение функции
    0.0074
```

Таким образом, приближенное значение корня равно 1,4304, с погрешностью 0,00032767. Значение функции равно 0,0074.

Т.е. из четырёх методов наилучшей сходимостью обладает метод Ньютона, что подтверждает теоретические данные по вычислительным методам.

Задание 5

Решение линейного дифференциального уравнения 2-го порядка

Решить линейное дифференциальное уравнение второго порядка с постоянными коэффициентами:

$$y''(t) + 5y'(t) + N_s y(t) = N_g, y(0) = 5, y'(0) = 10,$$

где $N_g = 28, N_s = 7$.

Задачу решить тремя различными методами:

- классическим аналитическим методом;
- операторным аналитическим методом;
- численным методом Рунге-Кутта 4-го порядка.

Сравнить полученные результаты, построить график решения.

Примечание. Для численного метода Рунге-Кутта предварительно требуется рассчитать время переходного процесса, используя найденные ранее корни характеристического уравнения. В качестве интервала наблюдения выбирается временной интервал $T = (3 \div 4)\tau$, где $\tau = 1/|p_{min}|$. Здесь p_{min} есть минимальный по величине (модулю) корень характеристического уравнения $p_{min} = \min\{|p_1|, |p_2|\}$.

Рекомендуемая величина шага интегрирования: $h = \Delta t = T/20$ для ручного счета, или $h = \Delta t = T/200$ при использовании программы метода Рунге-Кутта. Исходное уравнение 2-го порядка $y'' = f(y, y')$ преобразуется в систему двух уравнений 1-го порядка с помощью замены переменных:

$$\begin{cases} y' = z, y(0) = y_0, \\ z' = f(y, z), z(0) = y'(0) = y'_0. \end{cases}$$

Методом Рунге-Кутта решается данная система. Результаты заносятся в таблицу для построения графика решения $y(t)$.

Решение

Перепишем заданное уравнение в соответствии с данными варианта:

$$y''(t) + 5 \cdot y'(t) + 7 \cdot y(t) = 28, y(0) = 5, y'(0) = 10.$$

Краткое описание метода Рунге-Кутта 4 порядка

$y'' = f\left(x, y, y'\right)$ – дифференциальное уравнение второго порядка,

$\left(x_0, y_0, y_0'\right)$ – начальные условия

$$k_1^n = h \cdot f\left(x_n, y_n, y_n'\right),$$

$$k_2^n = h \cdot f\left(x_n + \frac{h}{2}, y_n + y_n' \frac{h}{2}, y_n' + \frac{k_1^n}{2}\right),$$

$$k_3^n = h \cdot f\left(x_n + \frac{h}{2}, y_n + y_n' \frac{h}{2} + h\frac{k_1^n}{2}, y_n' + \frac{k_2^n}{2}\right),$$

$$k_4^n = h \cdot f\left(x_n + h, y_n + y_n' h + k_2^n h / 2, y_n' + k_3^n\right),$$

$$y_{n+1}' = y_n' + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$$

$$y_{n+1} = y_n + h\left(y_n' + \frac{1}{6}\left(k_1 + k_2 + k_3\right)\right).$$

Для решения поставленной задачи в системе Matlab создаём 6 m-файлов, содержание которых представлено ниже на листингах 1 – 6.

Листинг 1. Файл основной программы **Difur_klas_oper_RK4.m**

```
% Решение дифференциального уравнения 2-го порядка
% классическим методом, операторным методом и
% методом Рунге-Кутты 4-го порядка
clear; % очистка памяти
clc;   % очистка рабочей области
a=0; % начало отрезка интегрирования
b=5;  % конец отрезка интегрирования
x0=[a;5;10]; %вектор начальных приближений
n=20; % число узлов разбиения интервала интегрирования
y=x0; % присвоение вектору решений начальных значений
% Решение по методу Рунге-Кутты 4-го порядка
y=Runge4(x0,a,b,n);
xx=y(1,:);
% Решение по методу Рунге-Кутта 4 порядка помощью
функции ode45 с
% автоматическим шагом
y10=[x0(2,1) x0(3,1)]; % вектор начальных условий
[x, Y] = ode45(@vdif, [a b], y10); % вызов функции
ode45
ymatlab=[x, Y]'; % решение с помощью функции ode45
xp=ymatlab(1,:);
```

```
yp=ymatlab(2,:);
% Решение по классическому методу
y1=klassica(xx);
% Решение по операторному методу
y2=oper(xx);
yop=y2';
disp('==========================================================')
disp('Таблица найденного решения методом Рунге-Кутта 4-го порядка')
disp('==========================================================')
fprintf(' k          t            y_runge      \n');
for k=1:1:n
    fprintf(' %d %14.7f %14.7f\n',k,xx(k),y(2,k));
end
disp('==========================================================')
disp('Таблица найденного решения классическим методом')
disp('==========================================================')
fprintf(' k          t            y_klassica      \n');
for k=1:1:n
    fprintf(' %d %14.7f %14.7f\n',k,xx(k),y1(k));
end
disp('==========================================================')
disp('Таблица найденного решения операторным методом')
disp('==========================================================')
fprintf(' k          t            y_oper      \n');
for k=1:1:n
    fprintf(' %d %14.7f %14.7f\n',k,xx(k),y2(k));
end
disp('==========================================================')
disp('Таблица найденного решения функцией matlab ode45')
disp('==========================================================')
fprintf(' k          t            y_matlab      \n');
for k=1:1:length(x)
    fprintf(' %d %14.7f %14.7f\n',k,xp(k),yp(k));
end
```

```
% Построение графиков решения дифференциального
уравнения первого порядка
% Метод Рунге-Кутта 4-го порядка
plot(xx,y(2,:),'r',xp,yp,'b-.',xx,y1,'x-.',xx,y2,'s');
grid on
hold on
legend('метод Рунге-Кутта 4 порядка','функция
ode45','классический метод','операторный метод')
ylabel('x') % название оси y
xlabel('t') % название оси x
title('График решения дифференциального уравнения
второго порядка')
```

Листинг 2. Файл функции для решения по методу Рунге-Кутта 4 порядка **fm.m**

```
function g=fm(x,y,z)
g=-5*z-7*y+28;
end
```

Листинг 3. Файл функции для решения по методу Рунге-Кутта 4 порядка при помощи функции Matlab – ode45 **vdif.m**

```
function dy=vdif(t,xx)
dy=zeros(2,1);
dy(1)=xx(2);
dy(2)=-5*xx(2)-7*xx(1)+28;
end
```

Листинг 4. Файл функция решения по методу Рунге-Кутта 4 порядка **Runge4.m**

```
function z=Runge4(x0,a,b,n)
% функция, реализующая метод Рунге-Кутты 4-го порядка
h=(b-a)/(n-1);
y=x0;
for i=1:n-1 %цикл поиска значений вектора решений по
методу Р-К
    % Вычисление 1-го коэффициента в методе Рунге-Кутта
    k(1,1)=h*fm(y(1,i),y(2,i),y(3,i));
    % Вычисление 2-го коэффициента в методе Рунге-Кутта

k(1,2)=h*fm(y(1,i)+h/2,y(2,i)+y(3,i)*h/2,y(3,i)+k(1,1)/
2);
   % Вычисление 3-го коэффициента в методе Рунге-Кутта
```

```
k(1,3)=h*fm(y(1,i)+h/2,y(2,i)+y(3,i)*h/2+h*k(1,1)/2,y(3
,i)+k(1,2)/2);
   % Вычисление 4-го коэффициента в методе Рунге-Кутта

k(1,4)=h*fm(y(1,i)+h,y(2,i)+y(3,i)*h+h*k(1,2)/2,y(3,i)+
k(1,3));
 %   %  нахождения начальных значений вектора решений

y(3,i+1)=y(3,i)+1/6*(k(1,1)+2*k(1,2)+2*k(1,3)+k(1,4));
% Метод Рунге-Кутта

y(2,i+1)=y(2,i)+h*(y(3,i)+1/6*(k(1,1)+k(1,2)+k(1,3)));
     y(1,i+1)=y(1,i)+h; % формирование элементов в 1-ой
строке в векторе решений(иксы)
end
z=y;
end
```

Листинг 5. Файл функция решения дифференциального уравнения классическим методом **klassica.m**

```
function z=klassica(t)
% функция, реализующая классический метод решения
% дифференциального уравнения
fklas=dsolve('D2y+5*Dy+7*y=28','y(0)=5','Dy(0)=10');
z=eval(fklas);
end
```

Листинг 6. Файл функция решения дифференциального уравнения операторным методом **oper.m**

```
function z=oper(t)
% функция, реализующая операторный метод решения
% дифференциального уравнения
% составляем уравнение с использованием операторов
Лапласа
syms Y p
Y=solve('p^2*Y-p*5-10+5*p*Y-5*5+7*Y=28/p','Y');
% обратное преобразование Лапласа
fop=ilaplace(Y,t);
z=eval(fop);
end
```

Описание программы

Программа разработана в среде MATLAB. В ней описаны методы решения дифференциального уравнения второго порядка: метод Рунге-Кутты 4-го порядка, классический метод и операторный метод.

Описание применения

Программа нахождения решения дифференциального уравнения второго порядка методами Рунге-Кутты 4-го порядка, классическим и операторным методами представлена в виде m – файла Difur_klas_oper_RK4.m. В ней определяются исходные данные: x0 – вектор начальных условий; a – начало отрезка интегрирования; b – конец отрезка интегрирования; n – число точек разбиения интервала интегрирования x. Запуск можно осуществить как в командном окне, набрав имя программы, так и из окна редактора m – файлов.

При выполнении программы в командном окне появляются соответствующие векторы решений для соответственных методов. Также выводится графическое окно, где иллюстрируются найденные решения соответствующими методами.

Результаты выполнения программы

```
=============================================================
Таблица найденного решения методом Рунге-Кутта 4-го порядка
=============================================================
 k          t            y_runge
 1      0.0000000      5.0000000
 2      0.2631579      6.2555776
 3      0.5263158      6.0508943
 4      0.7894737      5.4770951
 5      1.0526316      4.9410514
 6      1.3157895      4.5504488
 7      1.5789474      4.2997849
 8      1.8421053      4.1523449
 9      2.1052632      4.0716430
 10      2.3684211      4.0304186
 11      2.6315789      4.0109112
 12      2.8947368      4.0025546
 13      3.1578947      3.9995070
 14      3.4210526      3.9987531
 15      3.6842105      3.9988491
 16      3.9473684      3.9991656
 17      4.2105263      3.9994662
 18      4.4736842      3.9996868
 19      4.7368421      3.9998289
 20      5.0000000      3.9999128
=================================================
```

Таблица найденного решения классическим методом

k	t	y_klassica
1	0.0000000	5.0000000
2	0.2631579	6.1935897
3	0.5263158	5.9452803
4	0.7894737	5.3745169
5	1.0526316	4.8651739
6	1.3157895	4.5043123
7	1.5789474	4.2768059
8	1.8421053	4.1440357
9	2.1052632	4.0710892
10	2.3684211	4.0330965
11	2.6315789	4.0143270
12	2.8947368	4.0055787
13	3.1578947	4.0017861
14	3.4210526	4.0003058
15	3.6842105	3.9998294
16	3.9473684	3.9997458
17	4.2105263	3.9997893
18	4.4736842	3.9998556
19	4.7368421	3.9999108
20	5.0000000	3.9999487

Таблица найденного решения операторным методом

k	t	y_oper
1	0.0000000	5.0000000
2	0.2631579	6.1935897
3	0.5263158	5.9452803
4	0.7894737	5.3745169
5	1.0526316	4.8651739
6	1.3157895	4.5043123
7	1.5789474	4.2768059
8	1.8421053	4.1440357
9	2.1052632	4.0710892
10	2.3684211	4.0330965
11	2.6315789	4.0143270
12	2.8947368	4.0055787
13	3.1578947	4.0017861
14	3.4210526	4.0003058
15	3.6842105	3.9998294
16	3.9473684	3.9997458
17	4.2105263	3.9997893
18	4.4736842	3.9998556
19	4.7368421	3.9999108
20	5.0000000	3.9999487

Таблица найденного решения функцией matlab ode45

k	t	y_matlab
1	0.0000000	5.0000000
2	0.0088136	5.0859469
3	0.0176273	5.1676108
4	0.0264409	5.2451330
5	0.0352545	5.3186507
6	0.0793227	5.6308658
7	0.1233909	5.8614201
8	0.1674591	6.0235027
9	0.2115273	6.1288607
10	0.2574505	6.1892364
11	0.3033737	6.2083502
12	0.3492969	6.1942926
13	0.3952200	6.1541998

14	0.4411432	6.0941102
15	0.4870664	6.0187398
16	0.5329896	5.9321395
17	0.5789128	5.8378446
18	0.6309026	5.7254927
19	0.6828923	5.6101337
20	0.7348821	5.4942807
21	0.7868718	5.3800784
22	0.8438879	5.2586865
23	0.9009039	5.1425135
24	0.9579200	5.0324724
25	1.0149360	4.9293098
26	1.0783193	4.8232443
27	1.1417026	4.7261785
28	1.2050858	4.6379211
29	1.2684691	4.5582825
30	1.3370950	4.4813957
31	1.4057208	4.4134363
32	1.4743467	4.3536472
33	1.5429725	4.3013720
34	1.6167363	4.2528038
35	1.6905000	4.2111857
36	1.7642637	4.1756656
37	1.8380275	4.1455177
38	1.9175638	4.1182626
39	1.9971002	4.0956444
40	2.0766365	4.0769539
41	2.1561729	4.0615925
42	2.2437356	4.0479054
43	2.3312983	4.0369983
44	2.4188610	4.0283584
45	2.5064237	4.0215551
46	2.6114358	4.0153095
47	2.7164478	4.0107014
48	2.8214599	4.0073507
49	2.9264719	4.0049341
50	3.0071499	4.0035520
51	3.0878280	4.0024963
52	3.1685060	4.0017002
53	3.2491841	4.0011053
54	3.3298621	4.0006643
55	3.4105402	4.0003450
56	3.4912182	4.0001203
57	3.5718962	3.9999659
58	3.6427749	3.9998731
59	3.7136537	3.9998119
60	3.7845324	3.9997748
61	3.8554111	3.9997555
62	3.9160969	3.9997492
63	3.9767827	3.9997501
64	4.0374685	3.9997564
65	4.0981544	3.9997665
66	4.1612665	3.9997796
67	4.2243785	3.9997944
68	4.2874906	3.9998102
69	4.3506027	3.9998262
70	4.4213142	3.9998438
71	4.4920258	3.9998608
72	4.5627373	3.9998769
73	4.6334488	3.9998918
74	4.7134564	3.9999071
75	4.7934640	3.9999208
76	4.8734717	3.9999329
77	4.9534793	3.9999435

```
78        4.9651095        3.9999449
79        4.9767396        3.9999463
80        4.9883698        3.9999476
81        5.0000000        3.9999490
```

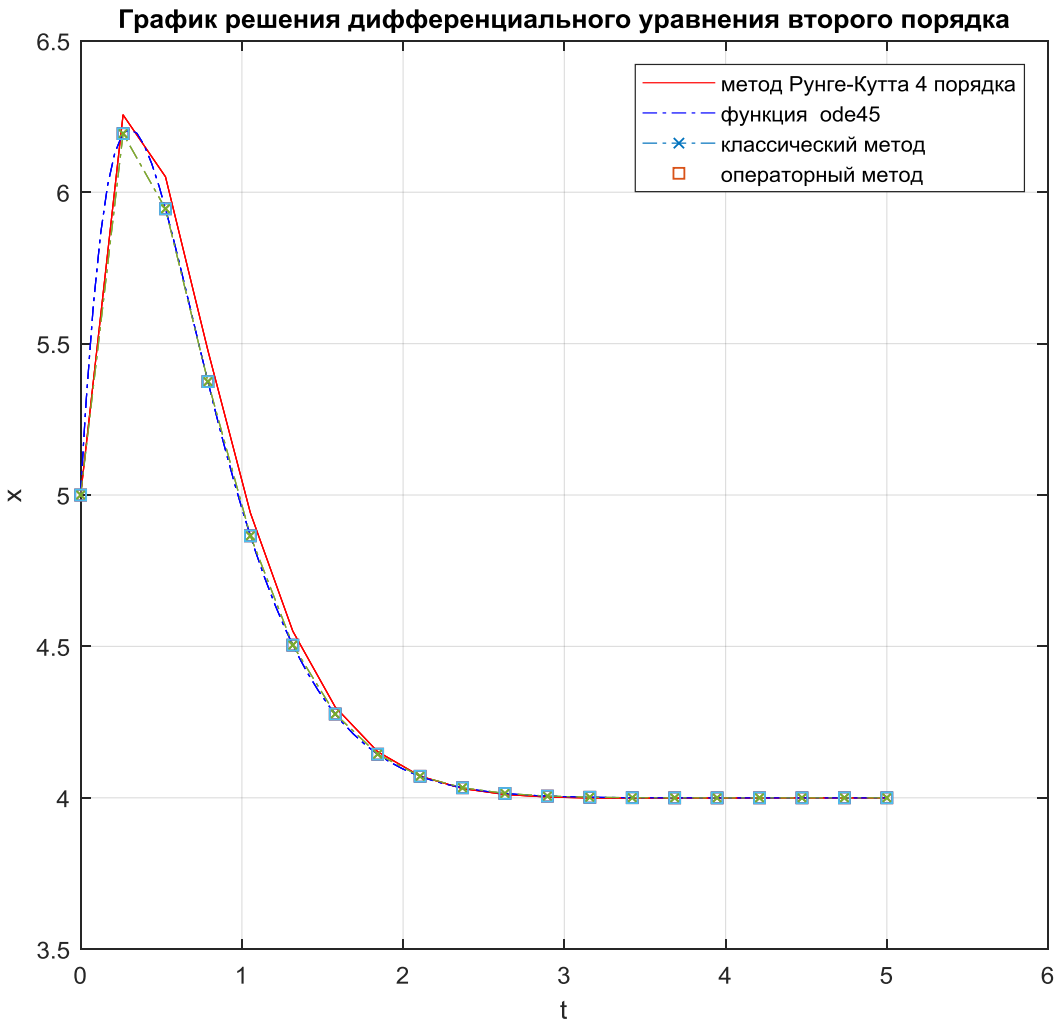

Рис. 4 – Визуализация решения дифференциального уравнения второго порядка разными методами

Как видно из результатов, полученных в командном окне программы, а также в графическом окне программы, результаты достаточно точно совпадают, следовательно, задача решена верно.

Задание 6

I. Интерполяция и аппроксимация

1. Выполнить интерполяцию функции $y(x)$, $x = [-5; -4; -1; 3; 7]$, $y = [3; 1; -4; 0; 3]$ используя функцию INTERP1 и методы CUBIC и SPLINE, сравнить результаты, построить графики с шагом 0,1.

Решение

Для решения поставленной задачи создаём m-файл, содержание которого отражено в листинге 1.

Листинг 1. Файл-сценарий intrpolyacia.m

```
% Интерполяция функции
clear; % очистка памяти
clc;   % очистка рабочей области
x = [-5 -4 -1 3 7];
y = [3 1 -4 0 3];
x2 = -5:0.1:7;
y1 = interp1(x,y,x2,'cubic');
y2 = interp1(x,y,x2,'spline');
plot(x,y,'*b',x2,y1,'-b',x2,y2,'-.r')
grid on
xlabel('x')
ylabel('y')
legend('заданные точки','cubic - интерполяция','spline
- интерполяция')
```

После запуска файла intrpolyacia.m на выполнение в графическом окне строится график (рис. 6.1).

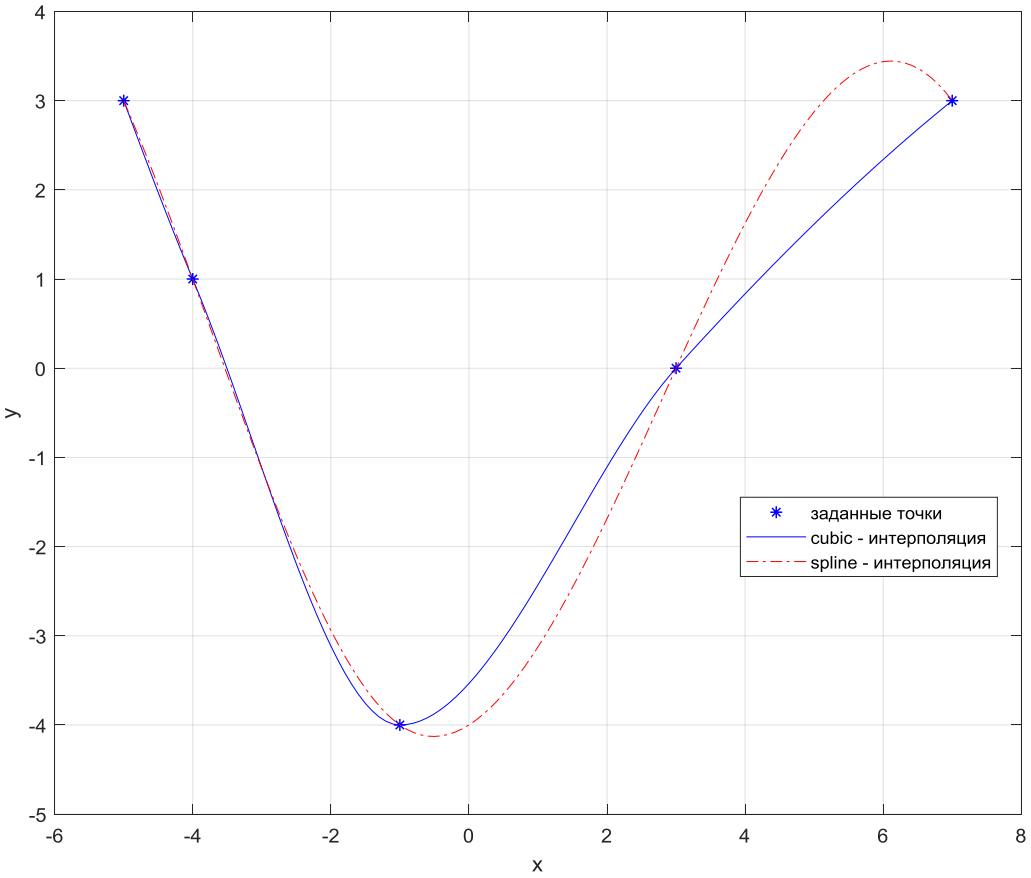

Рис. 6.1 – Результаты интерполяции разными методами

По рис. 6.1 можно сделать вывод, что интерполяция функции с использованием различных методов отличается.

Задание 7

II. Решение обыкновенных дифференциальных уравнений

4. Решение системы ОДУ (модель макроэкономической системы):

$$\begin{cases} x' = x \cdot y - a_1 \cdot x, \\ y' = b_1 \cdot x \cdot y - b_2 \cdot y^2 - b_3 \cdot y - b_4 \cdot (x - z), \\ z' = c_1 - c_2 \cdot x, \end{cases}$$

при $a_1 = 0{,}21$; $b_1 = 0{,}15$; $b_2 = 0{,}06$; $b_3 = 0{,}08$; $b_4 = 0{,}1$; $c_1 = 0{,}01$; $c_2 = 0{,}03$ и начальных условиях: $x(0) = 1$, $y(0) = 0{,}1$, $z(0) = 0{,}75$. Построить графики $x(t)$ – валовый продукт, $y(t)$ – темп роста валового продукта, $z(t)$ – платёжеспособный спрос и фазовый портрет системы на временном интервале [0;100].

Решение

Для решения поставленной задачи создаём m-файлы, содержание которых представлено на листингах 7.1 – 7.2.

Листинг 7.1. Файл-функция системы обыкновенных дифференциальных уравнений vdif.m

```
function dy=vdif(t,x)
a1=0.21;
b1=0.15;
b2=0.06;
b3=0.08;
b4=0.1;
c1=0.01;
c2=0.03;
dy=zeros(3,1);
dy(1)=x(1)*x(2)-a1*x(1);
dy(2)=b1*x(1)*x(2)-b2*x(2)^2-b3*x(2)-b4*(x(1)-x(3));
dy(3)=c1-c2*x(1);
end
```

Листинг 7.2. Файл-сценарий решения системы ОДУ sistema_ODU.m

```
% Решение системы ОДУ (модель макроэкономической
системы)
```

```
clear; % очистка памяти
clc;   % очистка рабочей области
a=0; % начало отрезка интегрирования
b=100;  % конец отрезка интегрирования
x0=[1;0.1;0.75]; % вектор начальных приближений
% Решение по методу Рунге-Кутта 4 порядка помощью
функции ode45 с
% автоматическим шагом
[tt, Y] = ode45(@vdif, [a b], x0); % вызов функции
ode45
y=[tt, Y]'; % решение с помощью функции ode45
t=y(1,:);
xp=y(2,:);
yp=y(3,:);
zp=y(4,:);
disp('=================================================
')
disp('Таблица найденного решения функцией ode45')
disp('=================================================
')
fprintf(' k          t            x          y
z \n');
for k=1:1:length(t)
    fprintf(' %d %14.7f %14.7f %14.7f
%14.7f\n',k,t(k),xp(k),yp(k),zp(k));
end
% Построение графиков решения дифференциального
уравнения первого порядка
% Метод Рунге-Кутта 4-го порядка
plot(t,xp,'r',t,yp,'b-.',t,zp,'x-.');
grid on
hold on
legend('x(t)-валовый продукт','y(t)-темп роста валового
продукта','z(t)-платёжеспособный спрос')
ylabel('x,y,z') % название оси y
xlabel('t') % название оси x
title('График решения системы ОДУ первого порядка')
```

После запуска файла sistema_ODU.m на выполнение получаем в командном окне следующие результаты:

```
=================================================
Таблица найденного решения функцией ode45
```

k	t	x	y	z
1	0.0000000	1.0000000	0.1000000	0.7500000
2	0.2700953	0.9700881	0.0952107	0.7447190
3	0.5401906	0.9399164	0.0909114	0.7396816
4	0.8102859	0.9096895	0.0871294	0.7348890
5	1.0803813	0.8795990	0.0838860	0.7303410
6	2.1443155	0.7656242	0.0765633	0.7147407
7	3.2082497	0.6638997	0.0777574	0.7025876
8	4.2721839	0.5785876	0.0866586	0.6934498
9	5.3361181	0.5110868	0.1018286	0.6867674
10	6.3291888	0.4630586	0.1200141	0.6822233
11	7.3222596	0.4277687	0.1407165	0.6789163
12	8.3153304	0.4037590	0.1627639	0.6764848
13	9.3084012	0.3896448	0.1851654	0.6746168
14	10.6414284	0.3843949	0.2144404	0.6725308
15	11.9744555	0.3938204	0.2414711	0.6703593
16	13.3074827	0.4174126	0.2650615	0.6675078
17	14.6405098	0.4552308	0.2843085	0.6634233
18	16.1292549	0.5143069	0.2997748	0.6567522
19	17.6180000	0.5919482	0.3077166	0.6469888
20	19.1067450	0.6855061	0.3068487	0.6333728
21	20.5954901	0.7865330	0.2958245	0.6154068
22	21.9435785	0.8735568	0.2762128	0.5952958
23	23.2916670	0.9365119	0.2468352	0.5721207
24	24.6397554	0.9577695	0.2087853	0.5471265
25	25.9878439	0.9284222	0.1657060	0.5222175
26	27.1940464	0.8593974	0.1272247	0.5018046
27	28.4002490	0.7620668	0.0934222	0.4844498
28	29.6064516	0.6517548	0.0679771	0.4709290
29	30.8126541	0.5431446	0.0527438	0.4614206
30	31.7929436	0.4642219	0.0480734	0.4564412
31	32.7732330	0.3961417	0.0496974	0.4536198
32	33.7535225	0.3396280	0.0566660	0.4526265
33	34.7338119	0.2939126	0.0678870	0.4531387
34	35.6113824	0.2608765	0.0806267	0.4546277
35	36.4889529	0.2343473	0.0952157	0.4568987
36	37.3665234	0.2133380	0.1110588	0.4597919
37	38.2440939	0.1970075	0.1276640	0.4631741
38	39.6991357	0.1784183	0.1558376	0.4695713
39	41.1541774	0.1683040	0.1836625	0.4765915
40	42.6092192	0.1650496	0.2101681	0.4838856
41	44.0642610	0.1679873	0.2347434	0.4911777
42	45.4593306	0.1765415	0.2561420	0.4979393
43	46.8544003	0.1908452	0.2751115	0.5042253
44	48.2494699	0.2114531	0.2914048	0.5097800
45	49.6445396	0.2391821	0.3047853	0.5143244
46	51.2710809	0.2816486	0.3163586	0.5179372
47	52.8976222	0.3369884	0.3230302	0.5191545
48	54.5241636	0.4059461	0.3240123	0.5173402
49	56.1507049	0.4868774	0.3183089	0.5118752
50	57.7772462	0.5755385	0.3049274	0.5022706
51	59.4037876	0.6596469	0.2825533	0.4883747
52	61.0303289	0.7232463	0.2509765	0.4707536
53	62.6568702	0.7497644	0.2118854	0.4508618
54	63.6960622	0.7405932	0.1845391	0.4379555
55	64.7352542	0.7111967	0.1571739	0.4256659
56	65.7744462	0.6645135	0.1315195	0.4145839
57	66.8136382	0.6052907	0.1091142	0.4051607
58	67.8528302	0.5398137	0.0913118	0.3976968
59	68.8920221	0.4737902	0.0788867	0.3922898
60	69.9312141	0.4114895	0.0720426	0.3888940
61	70.9704061	0.3558929	0.0705541	0.3873483
62	72.0846417	0.3051600	0.0741743	0.3874713

```
 63      73.1988773      0.2633580      0.0821755      0.3891368
 64      74.3131129      0.2298380      0.0935572      0.3920546
 65      75.4273485      0.2034814      0.1073855      0.3959712
 66      76.6698449      0.1810498      0.1246971      0.4012524
 67      77.9123413      0.1647144      0.1431551      0.4072533
 68      79.1548377      0.1533412      0.1620472      0.4137633
 69      80.3973341      0.1461392      0.1808425      0.4206150
 70      81.9290370      0.1422042      0.2033165      0.4293287
 71      83.4607399      0.1430918      0.2245229      0.4381125
 72      84.9924429      0.1485364      0.2440695      0.4467452
 73      86.5241458      0.1586391      0.2616784      0.4550185
 74      88.3134870      0.1768397      0.2795138      0.4639430
 75      90.1028283      0.2030047      0.2940406      0.4716814
 76      91.8921695      0.2384287      0.3048094      0.4777688
 77      93.6815108      0.2843288      0.3112744      0.4816787
 78      95.2611331      0.3341031      0.3128774      0.4828666
 79      96.8407554      0.3924248      0.3099604      0.4814700
 80      98.4203777      0.4571291      0.3018537      0.4771436
 81     100.0000000      0.5232246      0.2879443      0.4697128
>>
```

Также в отдельном графическом окне появляются результаты решения ОДУ (рис. 7.1).

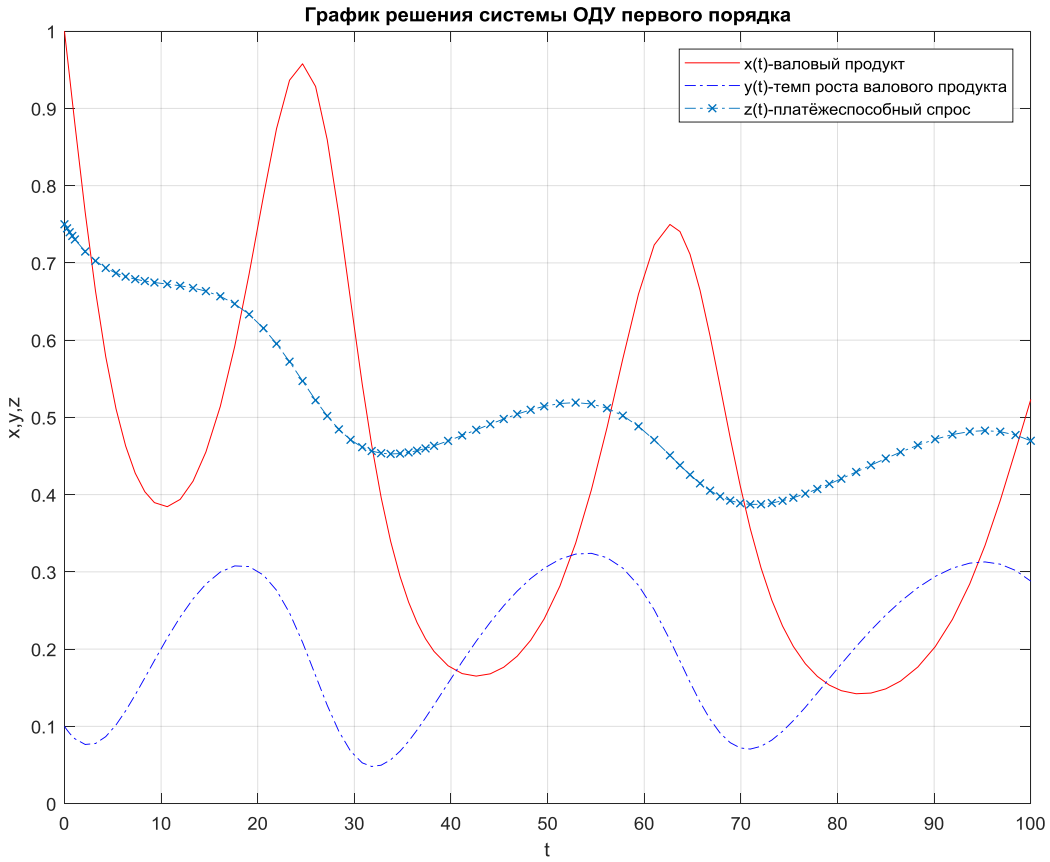

Рис. 7.1

Список литературы

1. Бахвалов Н.С. Численные методы в задачах и упражнениях: Учебное пособие / Н.С. Бахвалов, А.В. Лапин, Е.В. Чижонков. – М.: Бином, 2015.
2. Киреев В.И. Численные методы в примерах и задачах: Учебное пособие / В.И. Киреев, А.В. Пантелеев. – СПб.: Лань, 2015.
3. Самарский А.А. Введение в численные методы: Учебное пособие для вузов / А.А. Самарский. – СПб.: Лань, 2009.
4. Формалев В.Д. Численные методы / В.Д. Формалев, Д.Л. Ревизников. – М.: Физматлит, 2006.
5. Шахов Ю.Н. Численные методы / Ю.Н. Шахов, Е.И. Деза. – М.: КД Либроком, 2012.
6. Ширяев В.И. Исследование операций и численные методы оптимизации: Учебное пособие / В.И. Ширяев. – М.: Ленанд, 2015.